Data Preprocessing with Python

Machine Learning course, Chapter 3: Data Science, 9th Session
Project 2

SAMIRA SHEMIRANI #148

09/12/2023

Table of contents

01

Dataset

USA Market dataset

What's in the Dataset?

The dataset on which we intend to perform pre-processing is called the "USA Market" dataset.

- 112,457 data points have their information recorded in this dataset.
- 6 features have been examined for each data point.
- It can be seen that this dataset is a table with 112,457 rows and 8 columns.
- This dataset is 112,457 × 8.

The format of this file is ".csv", and here, in the next column, we name the features of each column:

1. Index
2. Date
3. Open
4. High
5. Low
6. Close
7. Adj Close
8. Volume

02 Methods

Features and Histograms.

How to code for preprocessing?

Features Description (x axis)

Index

This column shows us the type of index that we will only work with the NYA index in this project.

So many Indexes such as;
GDAXI, HSI, KS11, N100, NYA, TWII, SSMI, IXIC, ...

No Histogram Plot obtained.

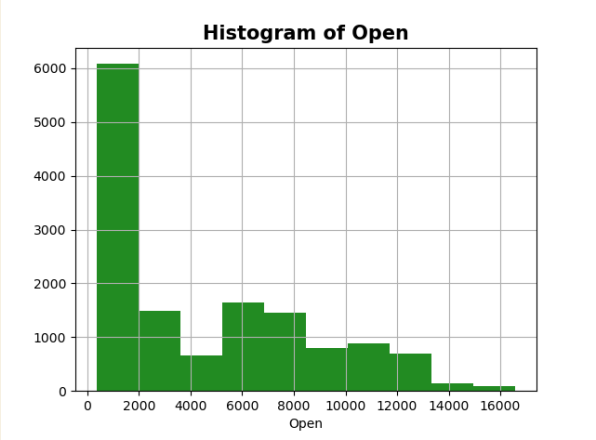
Date

This column shows the data registration date.
Starts from 1965 to 2005.

No Histogram Plot obtained.

Open

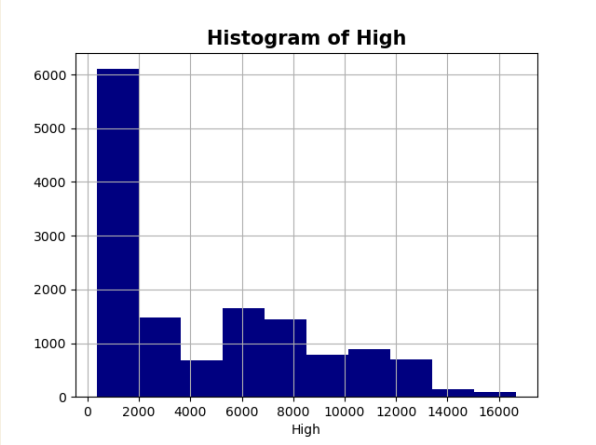
It is the price at which the financial security opens in the market when trading begins.
Starts from 347 to 11315.



Features Description (x axis)

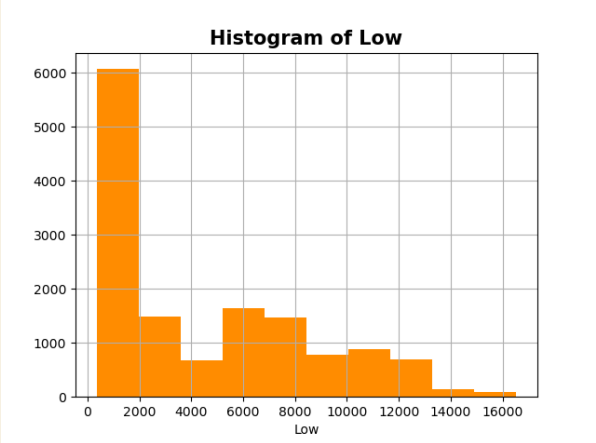
High

This column shows the highest price of that day.
Starts from 347 to 11334.



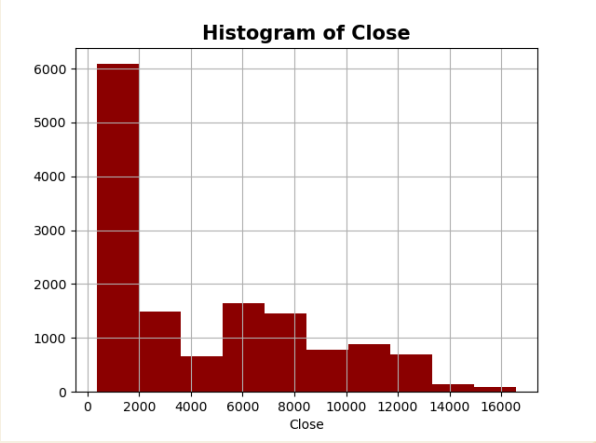
Low

This column shows the lowest price of that day.
Starts from 347 to 11069.



Close

At the close or closing price is the last traded price of securities like stocks, ETFs, and others at the end of regular market hours.
Starts from 347 to 11104.



Features Description (x axis)

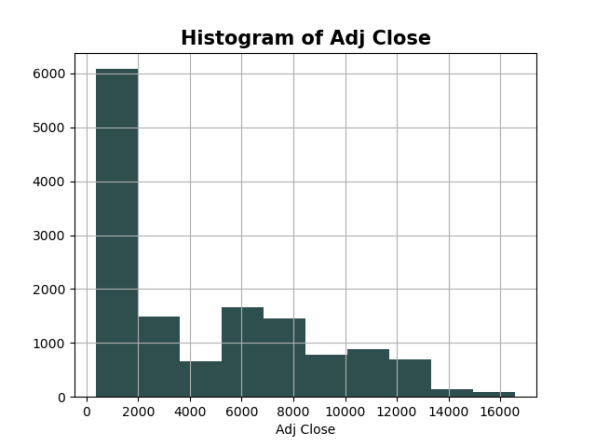
Volume

Volume is the amount of an asset or security that changes hands over some period of time, often over the course of a trading day.
In another way of saying this, volume is the number of shares traded.
Starts from 38,840,000 to 11,456,230,000.

Target Description (y axis)

Adj Close

The daily closing price, adjusted retroactively to include any corporate actions.



What we've done in the coding part?

Dataset

Reading the data and creating the DataFrame.

.describe()

Finding out some info's & if there is any MV's.

Missing Values

Finding MV's and removing them (.dropna()).

Plotting plots

Implementing plots codings

For the coding part, first we checked the dataset in Excel and Notepad++, and then we went to the Jupyter coding environment and did the following:

What's going on in our Jupyter Notebook?

Libraries

1 – We imported important libraries.

```
In [1]:
# importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Dataset Reading & DataFrame Creating

2 – We called the dataset into the coding environment, Jupyter Notebook & we made the DataFrame at the same time.

```
In [2]:
# reading the dataset file into the Jupyter-lab enviroment
data = pd.read_csv("D:/IMT/3- Data Science/9- Project #2/Market.csv")

# making the dataset 2D with the help of converting it to DataFrame
df = pd.DataFrame(data)
df
```

Out[2]:

	Index	Date	Open	High	Low	Close	Adj Close	Volume
0	NYA	12/31/1965	528.690002	528.690002	528.690002	528.690002	528.690002	0.0
1	NYA	1/3/1966	527.210022	527.210022	527.210022	527.210022	527.210022	0.0
2	NYA	1/4/1966	527.840027	527.840027	527.840027	527.840027	527.840027	0.0
3	NYA	1/5/1966	531.119995	531.119995	531.119995	531.119995	531.119995	0.0
4	NYA	1/6/1966	532.070007	532.070007	532.070007	532.070007	532.070007	0.0
...	...	...	...	...	...	...	...	...
112452	N100	5/27/2021	1241.119995	1251.910034	1241.119995	1247.069946	1247.069946	379696400.0

112457 rows × 8 columns

What's going on in our Jupyter Notebook?

Features & Target Recognition

3 – We decided which columns of Features are supposed to be our Target and which ones will remain as they are to be Features.

Features | Target | Must be removed

	Index	Date	Open	High	Low	Close	Adj Close	Volume
0	NYA	12/31/1965	528.690002	528.690002	528.690002	528.690002	528.690002	0.0
1	NYA	1/3/1966	527.210022	527.210022	527.210022	527.210022	527.210022	0.0
2	NYA	1/4/1966	527.840027	527.840027	527.840027	527.840027	527.840027	0.0
3	NYA	1/5/1966	531.119995	531.119995	531.119995	531.119995	531.119995	0.0
4	NYA	1/6/1966	532.070007	532.070007	532.070007	532.070007	532.070007	0.0
...	...	...	...	...	...	...	...	...
112452	N100	5/27/2021	1241.119995	1251.910034	1241.119995	1247.069946	1247.069946	379696400.0

Working only on “NYA”

4 – We removed every other rows except “NYA”s.

```
# finding the indexes that we want and saving it in "df1", we only want to work with the "NYA"s
df1 = df[df["Index"] == "NYA"]
df1
```

	Index	Date	Open	High	Low	Close	Adj Close	Volume
0	NYA	12/31/1965	528.690002	528.690002	528.690002	528.690002	528.690002	0.000000e+00
1	NYA	1/3/1966	527.210022	527.210022	527.210022	527.210022	527.210022	0.000000e+00
2	NYA	1/4/1966	527.840027	527.840027	527.840027	527.840027	527.840027	0.000000e+00
3	NYA	1/5/1966	531.119995	531.119995	531.119995	531.119995	531.119995	0.000000e+00
4	NYA	1/6/1966	532.070007	532.070007	532.070007	532.070007	532.070007	0.000000e+00
...	...	...	...	...	...	...	...	...
13943	NYA	5/24/2021	16375.000000	16508.519530	16375.000000	16464.689450	16464.689450	2.947400e+09
13944	NYA	5/25/2021	16464.689450	16525.810550	16375.150390	16390.189450	16390.189450	3.420870e+09
13945	NYA	5/26/2021	16390.189450	16466.339840	16388.320310	16451.960940	16451.960940	3.674490e+09
13946	NYA	5/27/2021	16451.960940	16546.359380	16451.960940	16531.949220	16531.949220	5.201110e+09
13947	NYA	5/28/2021	16531.949220	16588.689450	16531.949220	16555.660160	16555.660160	4.199270e+09

13948 rows × 8 columns

What's going on in our Jupyter Notebook?

Removing the “Volume” column

5- We don’t want to consider the “Volume” column, so let’s remove it.

```
# this is deleting the "Volume" column
df2 = df1.drop(columns = "Volume")
df2
```

13948 rows × 7 columns

Removing the decimal

6- In datapoints we have so many float numbers, which we don’t need. What we need is datapoints without any decimal part. So, let’s remove the decimal part of each datapoint.

```
# this is removing the decimal part of the numbers/data points
# keeping the data format and not changing it, but only rounding up/down each number
df3 = round(df2)
df3
```

	Index	Date	Open	High	Low	Close	Adj Close
0	NYA	12/31/1965	529.0	529.0	529.0	529.0	529.0
1	NYA	1/3/1966	527.0	527.0	527.0	527.0	527.0
2	NYA	1/4/1966	528.0	528.0	528.0	528.0	528.0
3	NYA	1/5/1966	531.0	531.0	531.0	531.0	531.0
4	NYA	1/6/1966	532.0	532.0	532.0	532.0	532.0
...	...	...	...	...	...	...	...
13943	NYA	5/24/2021	16375.0	16509.0	16375.0	16465.0	16465.0

What's going on in our Jupyter Notebook?

Sorting the DataFrame

7- Cause of having the “Date” columns, our DataFrame type is “Time Serie”, so let’s sort the rows based on the “Date” column.

```
df3.sort_values(by = ["Date"])
df3
```

Description and Information

8- Now that we have our DataFrame, let’s check out it’s description & information.

```
1 df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13948 entries, 0 to 13947
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Index      13948 non-null  object 
 1   Date       13948 non-null  object 
 2   Open       13947 non-null  float64
 3   High       13946 non-null  float64
 4   Low        13945 non-null  float64
 5   Close      13944 non-null  float64
 6   Adj Close  13938 non-null  float64
dtypes: float64(5), object(2)
memory usage: 871.8+ KB
```

What's going on in our Jupyter Notebook?

So how many Missing Values are there?

9-

```
1 df3.describe(include='all')
```

	Index	Date	Open	High	Low	Close	Adj Close
count	13948	13948	13947.000000	13946.000000	13945.000000	13944.000000	13938.000000
unique	1	13948	NaN	NaN	NaN	NaN	NaN
top	NYA	12/31/1965	NaN	NaN	NaN	NaN	NaN
freq	13948	1	NaN	NaN	NaN	NaN	NaN
mean	NaN	NaN	4452.144834	4469.313495	4434.260739	4453.026033	4455.093916
std	NaN	NaN	4074.832916	4094.959937	4052.814928	4075.484621	4075.457549
min	NaN	NaN	348.000000	348.000000	348.000000	348.000000	348.000000
25%	NaN	NaN	655.000000	655.000000	655.000000	655.000000	656.000000
50%	NaN	NaN	2632.000000	2632.000000	2632.000000	2632.000000	2633.000000
75%	NaN	NaN	7339.500000	7376.500000	7278.000000	7339.750000	7342.750000
max	NaN	NaN	16590.000000	16686.000000	16532.000000	16590.000000	16590.000000

There is 1 MV in the "Open": (Toppest numeber - 13947 = 1)
There are 2 MV's in the "High": (13948 - 13946 = 2)
There are 3 MV's in the "Low": (13948 - 13945 = 3)
There are 4 MV's in the "Close": (13948 - 13944 = 4)
There are 10 MV's in the "Adj Close": (13948 - 13938 = 10)

```
In [15]: 1 df3.isnull().sum()

Out[15]: Index        0
         Date         0
         Open         1
         High         2
         Low          3
         Close        4
         Adj Close   10
         dtype: int64
```

What's going on in our Jupyter Notebook?

What is the NaN row in all columns?

10-

```
df3[df3['Open'].isna()]
```

	Index	Date	Open	High	Low	Close	Adj Close
289	NYA	2/23/1967	NaN	NaN	NaN	NaN	NaN

```
df3[df3['High'].isna()]
```

	Index	Date	Open	High	Low	Close	Adj Close
190	NYA	9/30/1966	437.0	NaN	437.0	437.0	437.0
289	NYA	2/23/1967	NaN	NaN	NaN	NaN	NaN

```
df3[df3['Close'].isna()]
```

	Index	Date	Open	High	Low	Close	Adj Close
104	NYA	5/31/1966	492.0	492.0	492.0	NaN	492.0
170	NYA	9/1/1966	443.0	443.0	443.0	NaN	443.0
289	NYA	2/23/1967	NaN	NaN	NaN	NaN	NaN
464	NYA	11/1/1967	544.0	544.0	544.0	NaN	544.0

```
df3[df3['Low'].isna()]
```

	Index	Date	Open	High	Low	Close	Adj Close
102	NYA	5/26/1966	497.0	497.0	NaN	497.0	497.0
231	NYA	11/30/1966	461.0	461.0	NaN	461.0	461.0
289	NYA	2/23/1967	NaN	NaN	NaN	NaN	NaN

```
df3[df3['Adj Close'].isna()]
```

	Index	Date	Open	High	Low	Close	Adj Close
154	NYA	8/10/1966	477.0	477.0	477.0	477.0	NaN
257	NYA	1/9/1967	477.0	477.0	477.0	477.0	NaN
282	NYA	2/13/1967	507.0	507.0	507.0	507.0	NaN
289	NYA	2/23/1967	NaN	NaN	NaN	NaN	NaN
307	NYA	3/21/1967	521.0	521.0	521.0	521.0	NaN
333	NYA	4/27/1967	544.0	544.0	544.0	544.0	NaN
353	NYA	5/25/1967	532.0	532.0	532.0	532.0	NaN
635	NYA	7/19/1968	598.0	598.0	598.0	598.0	NaN
700	NYA	11/12/1968	623.0	623.0	623.0	623.0	NaN
800	NYA	4/16/1969	595.0	595.0	595.0	595.0	NaN

```
# we use this index of 289 from the table below,
# which was the only one and we got it earlier
nan_id = df3[df3['Open'].isna()].index

#nan_id = 289
```

```
df3_1 = df3.drop(nan_id)
df3_1
```

Row 289 will be removed.

	Index	Date	Open	High	Low	Close	Adj Close
0	NYA	12/31/1965	529.0	529.0	529.0	529.0	529.0
1	NYA	1/3/1966	527.0	527.0	527.0	527.0	527.0
2	NYA	1/4/1966	528.0	528.0	528.0	528.0	528.0
3	NYA	1/5/1966	531.0	531.0	531.0	531.0	531.0
4	NYA	1/6/1966	532.0	532.0	532.0	532.0	532.0
...	...	...	...	...	...	...	...
13943	NYA	5/24/2021	16375.0	16509.0	16375.0	16465.0	16465.0
13944	NYA	5/25/2021	16465.0	16526.0	16375.0	16390.0	16390.0
13945	NYA	5/26/2021	16390.0	16466.0	16388.0	16452.0	16452.0
13946	NYA	5/27/2021	16452.0	16546.0	16452.0	16532.0	16532.0
13947	NYA	5/28/2021	16532.0	16589.0	16532.0	16556.0	16556.0

13947 rows × 7 columns

We remove this NaN row.

What's going on in our Jupyter Notebook?

Decide what to do with Missing Values?

“.fillna()” or “.dropna()”?

11- For filling in the NaN values, we do:

```
# filling in missing values
df3_2 = df3_1.fillna(method='ffill', axis=1)
df3_2
```

```
In [25]: # LET'S CHECK THIS OUT AGAIN:
         # sum of null values in each columns of our DataFrame,
         # can be shown as:
         df3_2.isnull().sum()

Out[25]: Index        0
         Date         0
         Open         0
         High         0
         Low          0
         Close        0
         Adj Close    0
         dtype: int64
```

As you can see, there is no Missing Value left, they're all filled!

What's going on in our Jupyter Notebook?

".fillna()" or ".dropna()"?

12- For deleting in the NaN values as below.

```
# in here we implement "dropna()" for the whole DataFrame of us
df4 = df3_1.dropna()
df4
```

	Index	Date	Open	High	Low	Close	Adj Close
0	NYA	12/31/1965	529.0	529.0	529.0	529.0	529.0
1	NYA	1/3/1966	527.0	527.0	527.0	527.0	527.0
2	NYA	1/4/1966	528.0	528.0	528.0	528.0	528.0
3	NYA	1/5/1966	531.0	531.0	531.0	531.0	531.0
4	NYA	1/6/1966	532.0	532.0	532.0	532.0	532.0
...	...	...	...	...	...	...	...
13943	NYA	5/24/2021	16375.0	16509.0	16375.0	16465.0	16465.0
13944	NYA	5/25/2021	16465.0	16526.0	16375.0	16390.0	16390.0
13945	NYA	5/26/2021	16390.0	16466.0	16388.0	16452.0	16452.0
13946	NYA	5/27/2021	16452.0	16546.0	16452.0	16532.0	16532.0
13947	NYA	5/28/2021	16532.0	16589.0	16532.0	16556.0	16556.0

13932 rows × 7 columns

The shape of the DataFrame has changed because with dropna() you remove the rows which have NaN values: The shape used to be (13947, 7).

```
In [29]: 1 df4.shape
Out[29]: (13932, 7)
```

03

Plots

Scatter plot,
Line chart,
Count plot,
and
Pair plot.

What is a Scatter plot?

A scatterplot shows the relationship between two quantitative variables measured for the same individuals.

The values of one variable appear on the horizontal axis, and the values of the other variable appear on the vertical axis.

Each individual in the data appears as a point on the graph.

What is a Line chart?

A line chart or line graph, also known as curve chart, is a type of chart which displays information as a series of data points called 'markers' connected by straight line segments.

It is a basic type of chart common in many fields.

What is a Count plot?

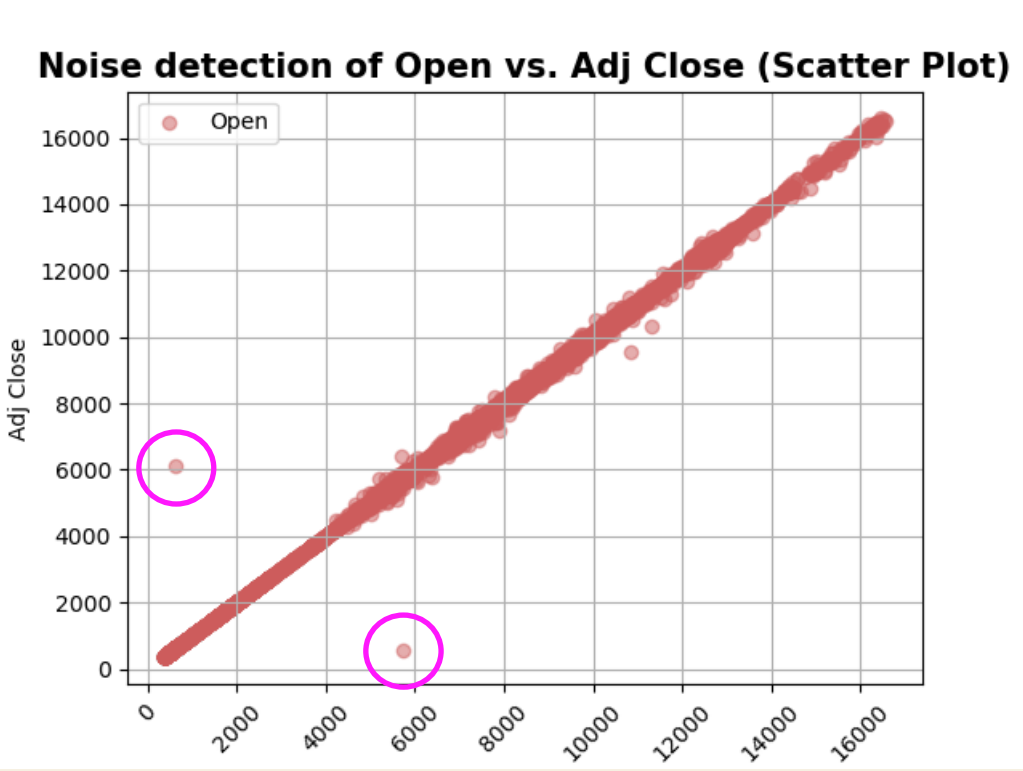
Show the counts of observations in each categorical bin using bars.

A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable.

To answer what is the difference between Barplot and Countplot in Seaborn, we have to say that, countplot plots the count of the number of records by category while barplot plots a value or metric for each category (by default, barplot plots the mean of a variable, by category)

Error Checking
Open vs. Adj Close

- As can be seen, the drawing of these two columns relative to each other shows the existence of two values out of the routine.
- Now we need to specify the location of these two values with correct addressing and perform the next action.
- We should be aware that we have to first ask the collector of this dataset about these two values, if she/he has doubts and there is no convincing reason to keep these two values outside the range, then we will definitely consider these two values as noise and take action.
- And that action would be removing the noises.



Error Checking
Open vs. Adj Close (cont.)

- Now you can see the steps to detect the index of data points that are considered noise:

In [36]:
```
# implementing the address to find out the noise index => result: The index is 831
Noise_up = df4[ (df4["Adj Close"]>5000) & (df4["Open"]<2000) ]
Noise_up
```

Out[36]:

	Index	Date	Open	High	Low	Close	Adj Close
831	NYA	5/29/1969	612.0	612.0	612.0	612.0	6111.0

In [37]:
```
# implementing the address to find out the other noise index => result: The index is 852
Noise_down = df4[ (df4["Adj Close"]<1500) & (df4["Open"]>5000) ] # (df4["Adj Close"]<2000)
Noise_down
```

Out[37]:

	Index	Date	Open	High	Low	Close	Adj Close
852	NYA	6/30/1969	5722.0	572.0	572.0	572.0	572.0

Now that I have these two indexes, I can remove these two noises by drop() command.
But before that, let's see what will happen if we use the Single Condition instead of the Binary Condition?!

- And here is the process of removing these two indexes (noises):

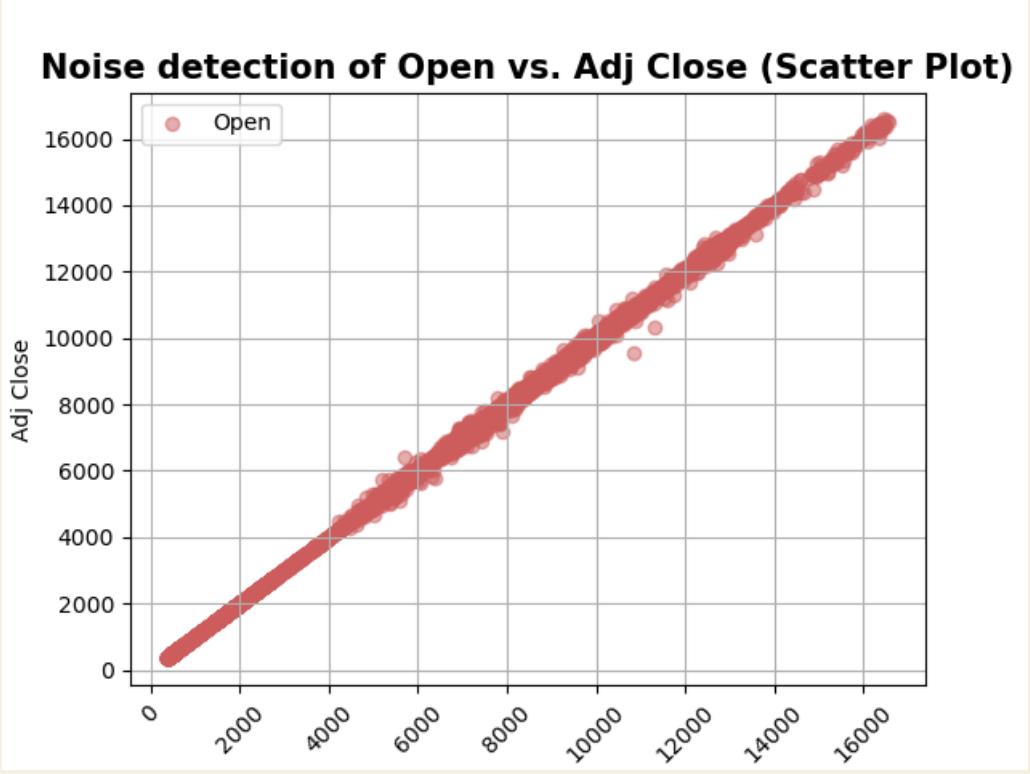
```
df5 = df4.drop(index = [831, 852])
df5
```

	Index	Date	Open	High	Low	Close	Adj Close
0	NYA	12/31/1965	529.0	529.0	529.0	529.0	529.0
1	NYA	1/3/1966	527.0	527.0	527.0	527.0	527.0
2	NYA	1/4/1966	528.0	528.0	528.0	528.0	528.0
3	NYA	1/5/1966	531.0	531.0	531.0	531.0	531.0
4	NYA	1/6/1966	532.0	532.0	532.0	532.0	532.0
...	...	...	...	...	...	...	...
13943	NYA	5/24/2021	16375.0	16509.0	16375.0	16465.0	16465.0
13944	NYA	5/25/2021	16465.0	16526.0	16375.0	16390.0	16390.0
13945	NYA	5/26/2021	16390.0	16466.0	16388.0	16452.0	16452.0
13946	NYA	5/27/2021	16452.0	16546.0	16452.0	16532.0	16532.0
13947	NYA	5/28/2021	16532.0	16589.0	16532.0	16556.0	16556.0

13930 rows × 7 columns

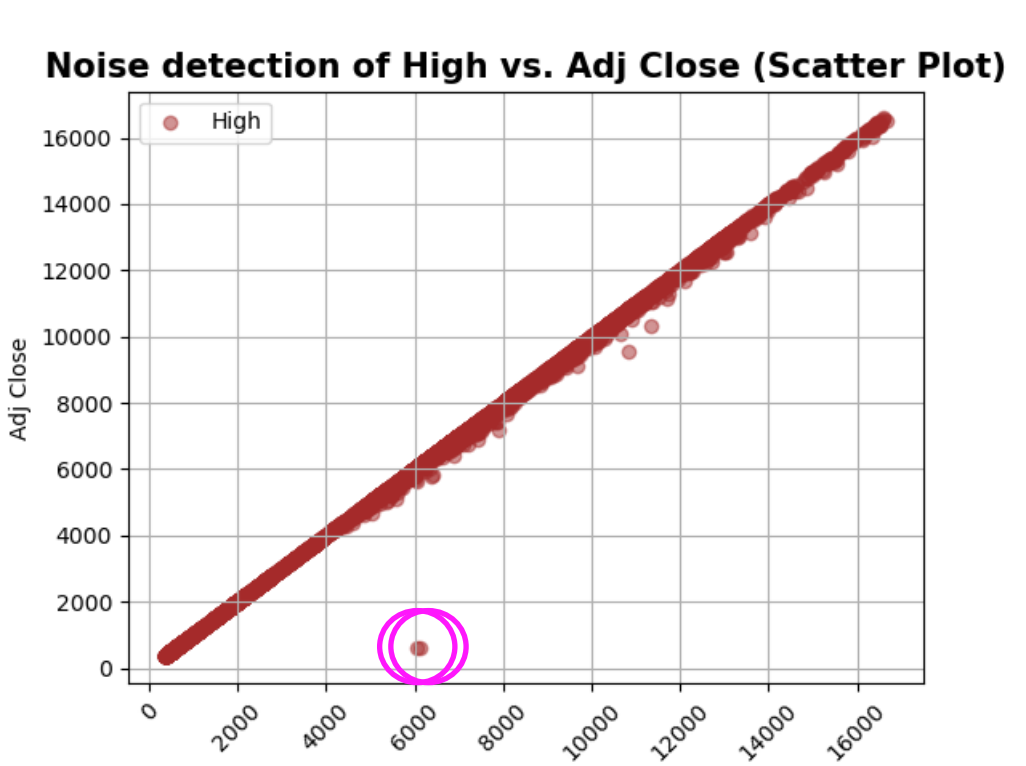
Error Checking
Open vs. Adj Close (cont.)

- Let’s check out the situation!
- Noises are gone!



Error Checking
High vs. Adj Close

- Now, check this two columns.



- How many indexes are there?

```
# implementing the address to find out the noise index => result: The index are 829 & 833
Noise_down_2 = df5[ (df5["Adj Close"]<2000) & (df5["High"]>5000) ]
Noise_down_2
```

	Index	Date	Open	High	Low	Close	Adj Close
829	NYA	5/27/1969	612.0	6124.0	612.0	612.0	612.0
833	NYA	6/3/1969	607.0	6066.0	607.0	607.0	607.0

Now is the time to remove these two achieved indexes! Using .drop()
INDEXES are 829 & 833

- There are two indexes which have to be removed.

Error Checking
High vs. Adj Close (cont.)

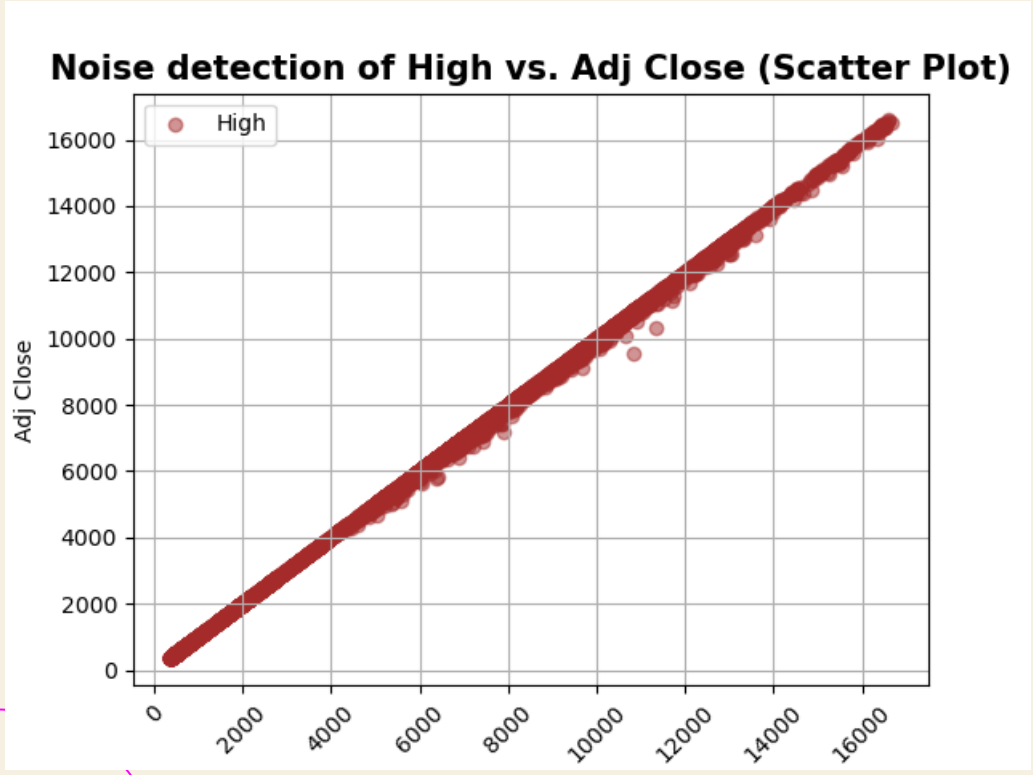
- Removing these two indexes:

```
df6 = df5.drop(index = [829, 833])
df6
```

	Index	Date	Open	High	Low	Close	Adj Close
0	NYA	12/31/1965	529.0	529.0	529.0	529.0	529.0
1	NYA	1/3/1966	527.0	527.0	527.0	527.0	527.0
2	NYA	1/4/1966	528.0	528.0	528.0	528.0	528.0
3	NYA	1/5/1966	531.0	531.0	531.0	531.0	531.0
4	NYA	1/6/1966	532.0	532.0	532.0	532.0	532.0
...	...	...	...	...	...	...	...
13943	NYA	5/24/2021	16375.0	16509.0	16375.0	16465.0	16465.0
13944	NYA	5/25/2021	16465.0	16526.0	16375.0	16390.0	16390.0
13945	NYA	5/26/2021	16390.0	16466.0	16388.0	16452.0	16452.0
13946	NYA	5/27/2021	16452.0	16546.0	16452.0	16532.0	16532.0
13947	NYA	5/28/2021	16532.0	16589.0	16532.0	16556.0	16556.0

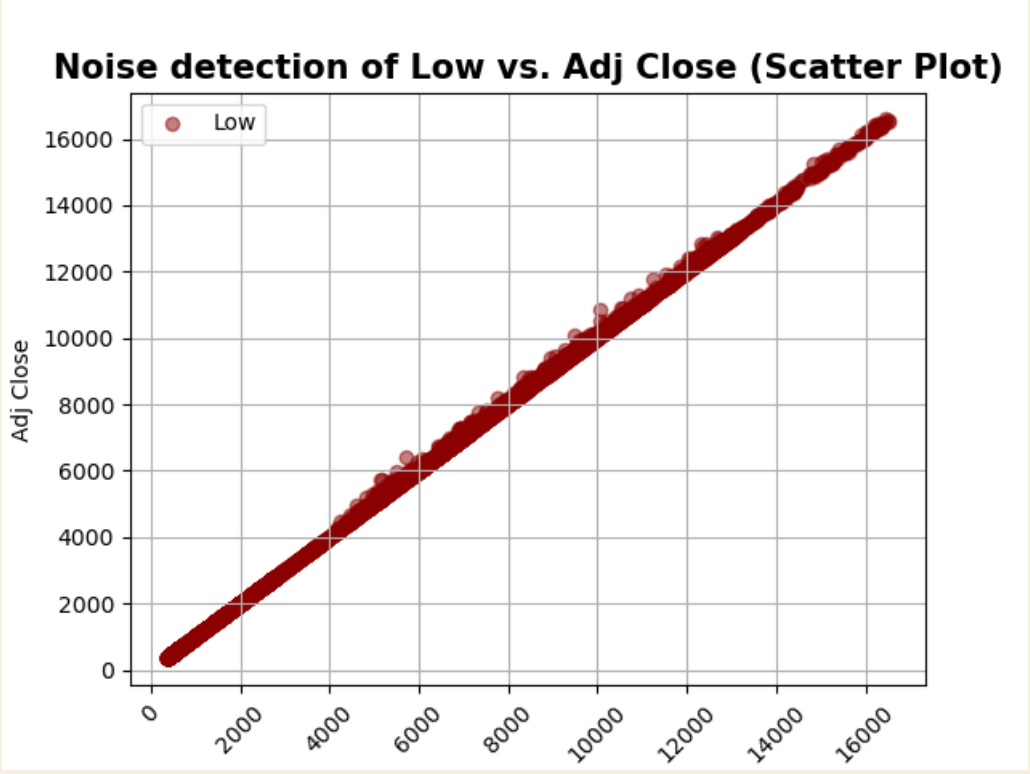
13928 rows × 7 columns

- Let's look at the plot again: (IT IS CLEAN OF NOISE NOW!)



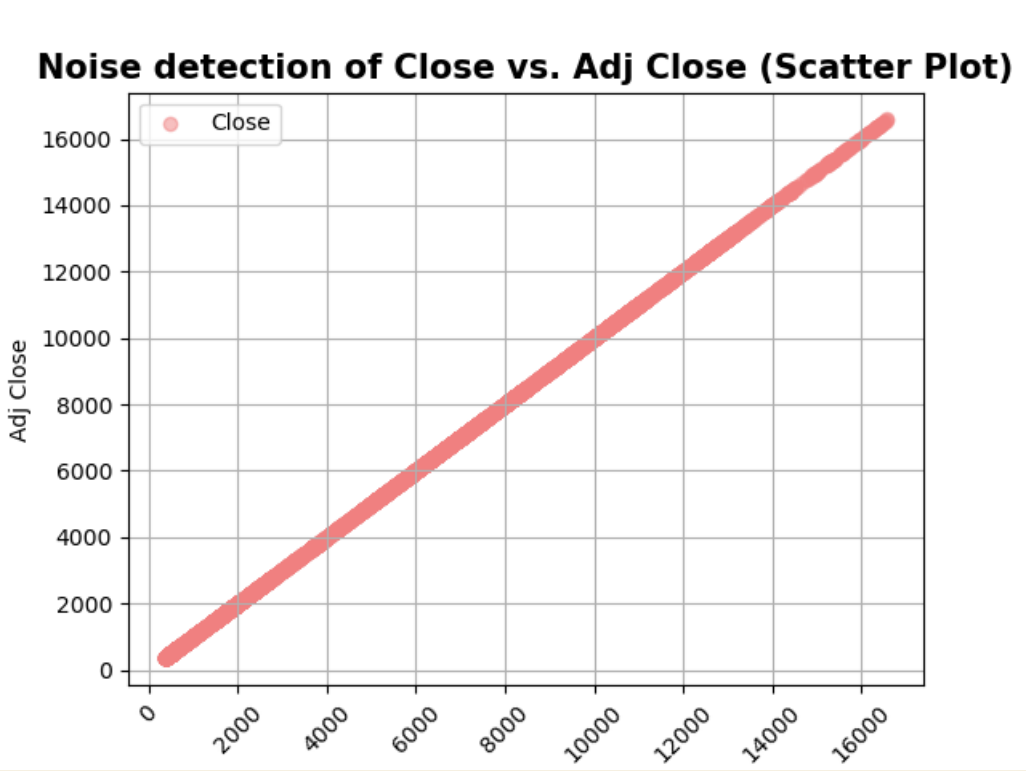
Error Checking
Low vs. Adj Close

- This plot seems to be so clean!



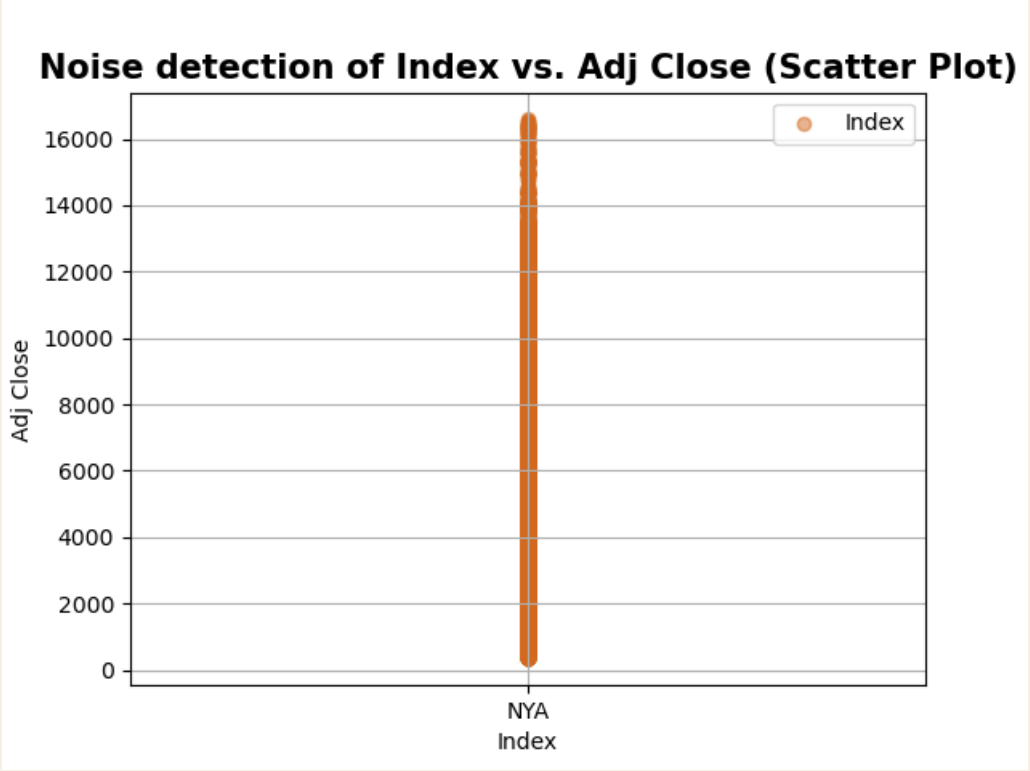
Error Checking
Close vs. Adj Close

- This plot also seems to be so clean, too!



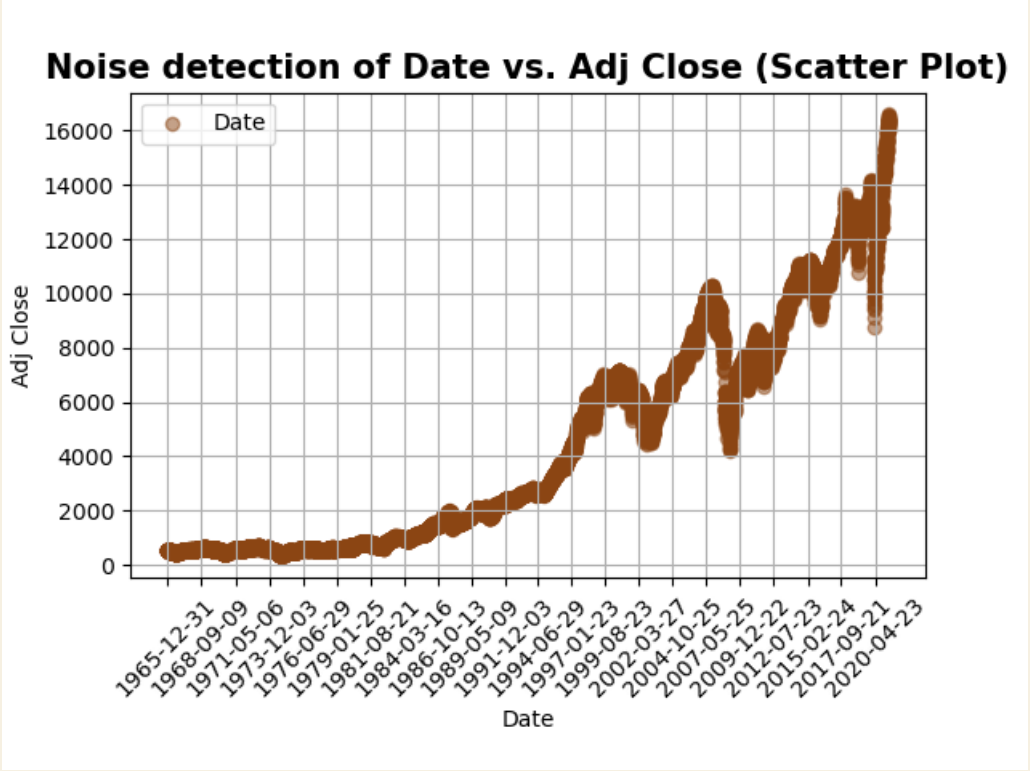
Error Checking
Index vs. Adj Close

- Also this plot;



Error Checking
Date vs. Adj Close

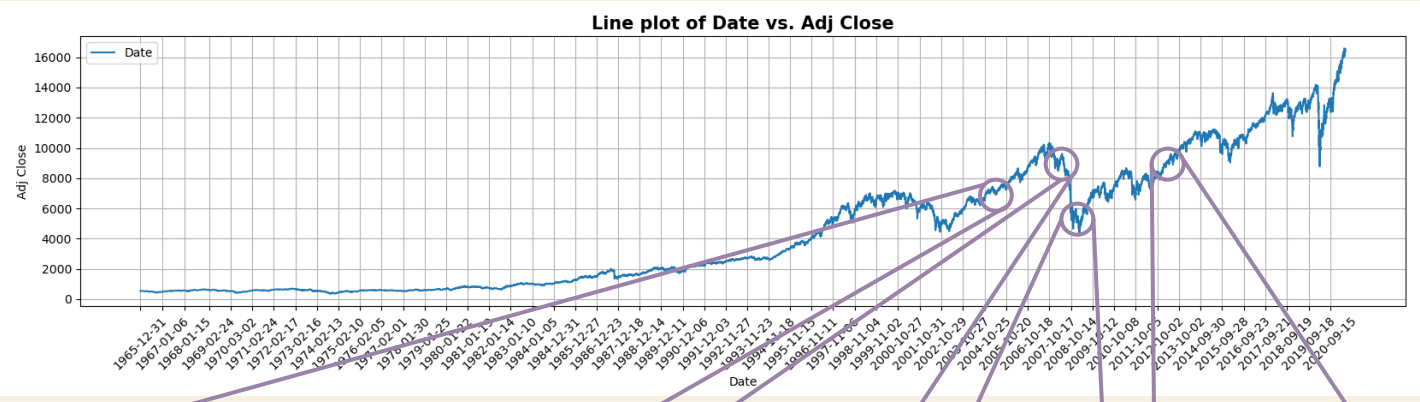
- And finally also this one;



Line Chart
Date vs. Adj Close

This important chart!

- This is the ***price-time*** chart.
- On the x-axis you can see the “Date” from 1965 to 2021.
- You can see the “Adj Close” in the y-axis.
- According to this chart, as date passes, the price increases on average.
- Of course, if you check each year more closely, you will see significant increases and decreases during each year like the four magnifier in below.
- In the next page you have this chart much bigger in vertical position.

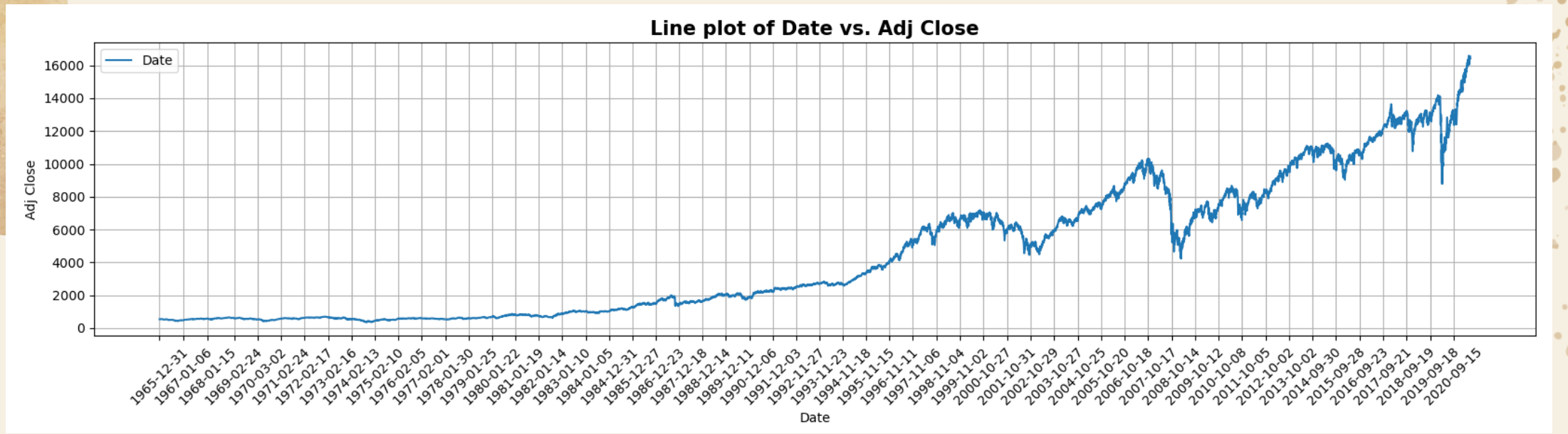


Ascending

Descending

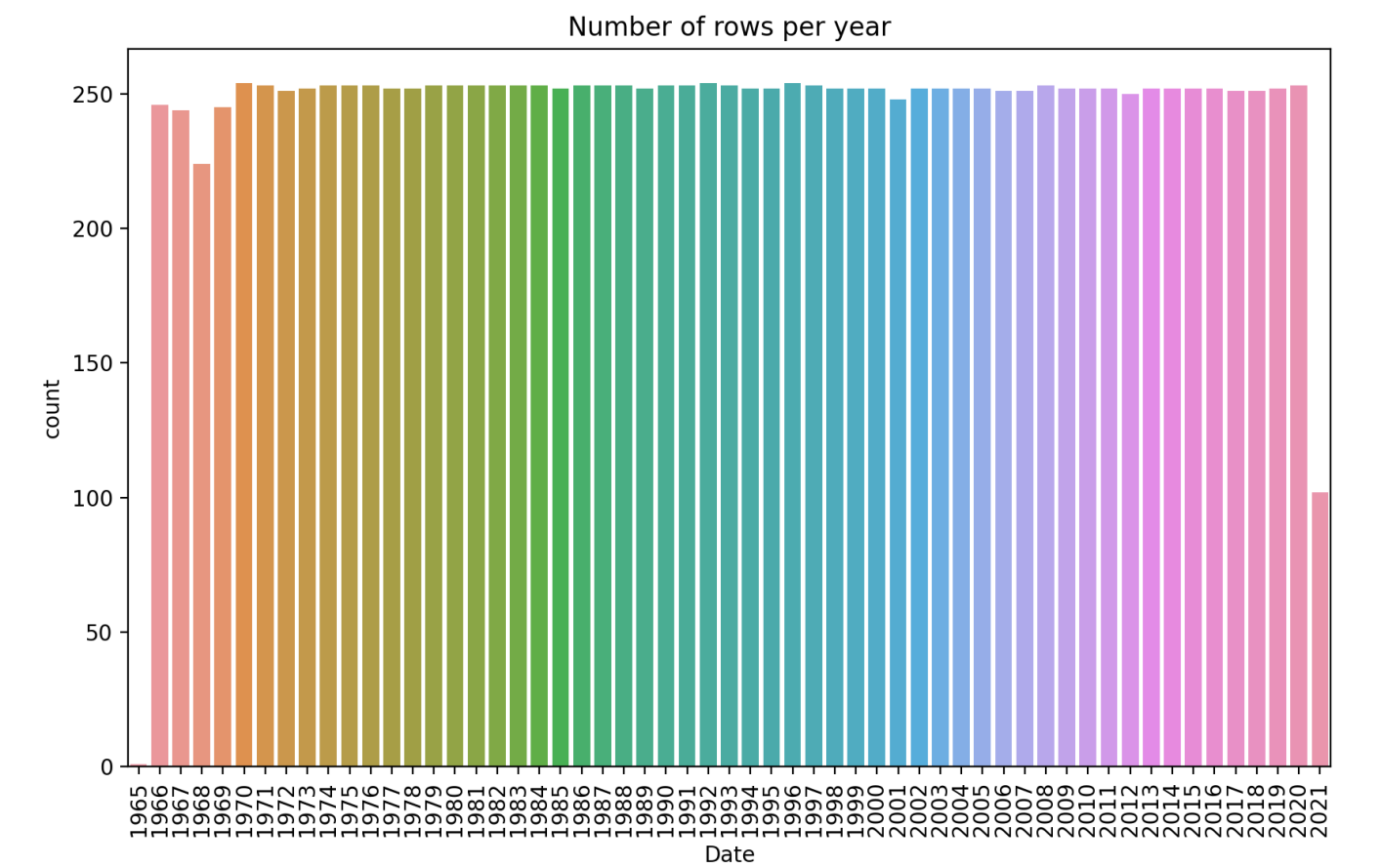
2007–2008 Financial Crisis

Ascending


Line plot of Date vs. Adj Close
Date
Adj Close
16000
14000
12000
10000
8000
6000
4000
2000
0
1965-12-31
1967-01-06
1968-01-15
1969-02-24
1970-03-02
1971-02-24
1972-02-17
1973-02-16
1974-02-13
1975-02-10
1976-02-05
1977-02-01
1978-01-30
1979-01-25
1980-01-22
1981-01-19
1982-01-14
1983-01-10
1984-01-05
1984-12-31
1985-12-27
1986-12-23
1987-12-18
1988-12-14
1989-12-11
1990-12-06
1991-12-03
1992-11-27
1993-11-23
1994-11-18
1995-11-15
1996-11-11
1997-11-06
1998-11-04
1999-11-02
2000-10-27
2001-10-31
2002-10-29
2003-10-27
2004-10-25
2005-10-20
2006-10-18
2007-10-17
2008-10-14
2009-10-12
2010-10-08
2011-10-05
2012-10-02
2013-10-02
2014-09-30
2015-09-28
2016-09-23
2017-09-21
2018-09-19
2019-09-18
2020-09-15

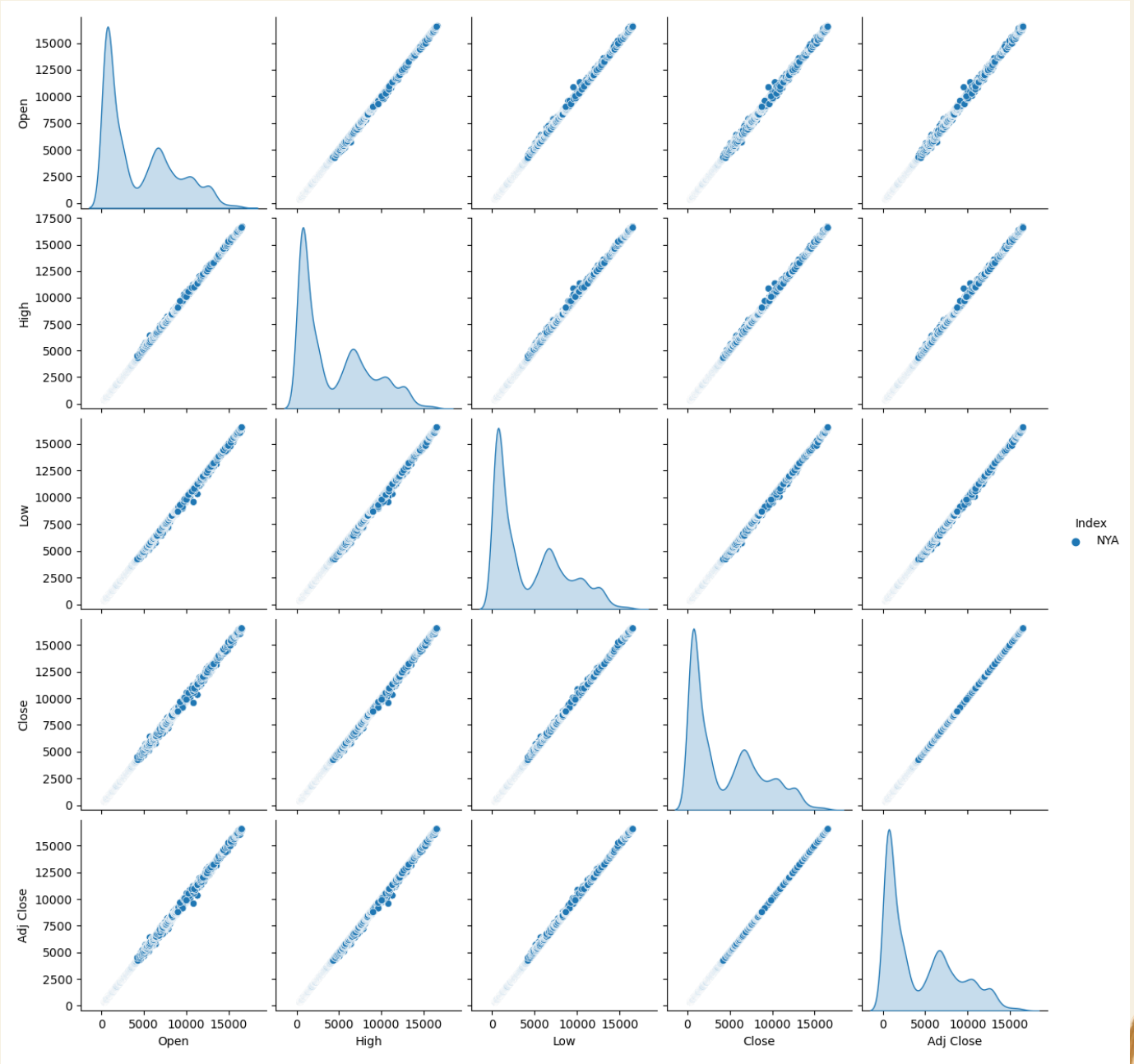
Count Plot by Seaborn

- Here you can see the number of data points according to each year.



Pair Plot
Index vs. all other columns

- Here is an extra plot to get more detail at the same time.



04

Conclusion

What was done.

What was achieved.

Summary & Conclusion

In this report, the USA Market dataset was examined, necessary pre-processing and error checking was done and plots were drawn.
Plots were based on the comparison of Feature and Target.
Each of these plots showed us relationships between samples, data points, and data within the dataset. Relationships that were not possible to discover in normal mode and only by looking at the table called dataset. These relationships give the reader of the report, even if he/she does not have expertise and knowledge about this dataset, valuable and categorized information.
The purpose of plotting these data is to discover the relationships between the data.

References

For describing Histograms:

Histograms review (article) | Khan Academy

For describing Scatter plot:

Describing scatterplots (form, direction, strength, outliers) (article) | Khan Academy

The end

Do you have any questions?

Samira.Shemirani92@gmail.com

www.linkedin.com/in/samira-shemirani-664302132