prediction analysis on stock_dataset ineuron internship

In [8]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import keras as k
import sklearn as skl
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import accuracy_score
```

In [9]:

```
df=pd.read_csv("nasdaq_data.csv")
```

In [10]:

```
df.head()
```

Out[10]:

	Symbol
0	AACI
1	AAME
2	AAOI
3	ABTX
4	ACAB

In [11]:

```
df.tail()
```

Out[11]:

	Symbol
673	YJ
674	ZING
675	ZLAB
676	ZNTL
677	ZT

In [12]:

```
df.columns
```

Out[12]:

```
Index(['Symbol', 'Name', 'Last Sale', 'Net Change', '% Change',
'Market Cap',
       'Country', 'IPO Year', 'Volume', 'Sector', 'Industry'],
      dtype='object')
```

In [13]:

```
df.info
```

Out[13]:

```
<bound method DataFrame.info of     Symbol
Name Last Sale  \
0     AACI            Armada Acquisition Corp. I Common Stock
$9.88
1     AAME         Atlantic American Corporation Common Stock
$2.99
2     AAOI          Applied Optoelectronics Inc. Common Stock
$2.25
3     ABTX            Allegiance Bancshares Inc. Common Stock
$40.55
4     ACAB  Atlantic Coastal Acquisition Corp. II Class A ...
$9.98
..     ...                                                ...       ..
.
673     YJ              Yunji Inc. American Depository Shares
$1.0607
674   ZING   FTAC Zeus Acquisition Corp. Class A Common Stock
$9.83
675   ZLAB         Zai Lab Limited American Depositary Shares
$30.39
676   ZNTL         Zentalis Pharmaceuticals Inc. Common Stock
$27.02
677     ZT  Zimmer Energy Transition Acquisition Corp. Cla...
$9.76

     Net Change % Change  Market Cap       Country  IPO Year   Volume
\
0       -0.0100   -0.10%   204609860  United States    2021.0     4452

1       -0.0200   -0.66%    61006692  United States       NaN     1205

2       -0.0600   -2.60%    62176685  United States    2013.0   197324

3       -0.1100   -0.27%   826330090  United States    2015.0    37469
```

```
4        0.0000    0.00%   299400000  United States    2022.0     1100

..          ...      ...          ...             ...       ...      ...

673     -0.0143   -1.33%   227711117         China    2019.0   117377

674     -0.0200   -0.20%   550856813  United States    2022.0     5028

675      1.6900    5.89%  2290661779         China    2017.0  1276727

676      2.0300    8.12%  1539053904  United States    2020.0   973057

677      0.0000    0.00%   420900000  United States    2021.0        3


                     Sector
Industry
0               Industrials             Consumer
Electronics/Appliances
1                   Finance                              Life
Insurance
2                Technology
Semiconductors
3                   Finance                                 Major
Banks
4               Industrials             Consumer
Electronics/Appliances
..                      ...                                           ..
.
673  Consumer Discretionary                      Other Specialty
Stores
674             Industrials             Consumer
Electronics/Appliances
675             Health Care  Biotechnology: Pharmaceutical
Preparations
676  Consumer Discretionary                         Specialty
Chemicals
677             Industrials             Consumer
Electronics/Appliances

[678 rows x 11 columns]>
```

In [14]:

```
df.isnull().sum()
```

Out[14]:

```
Symbol          0
Name            0
Last Sale       0
Net Change      0
% Change        0
Market Cap      0
Country         1
IPO Year      163
Volume          0
Sector        117
Industry      117
dtype: int64
```

In [15]:

```
df.shape
```

Out[15]:

```
(678, 11)
```

In [16]:

```
len(df.Symbol.unique())
```

Out[16]:

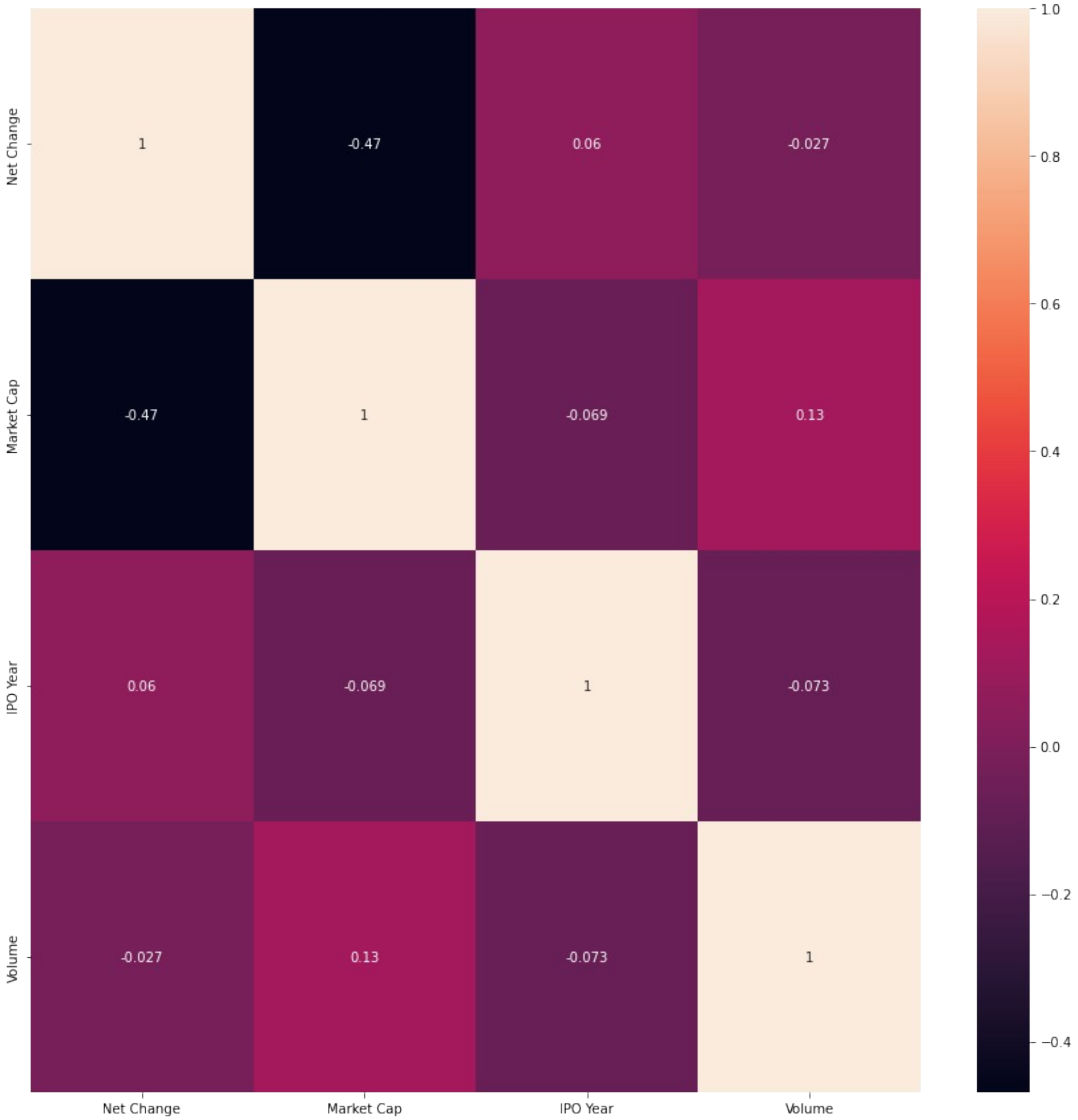
```
678
```

In [17]:

```
corr = df.corr()
plt.figure(figsize = (15 ,15))
sns.heatmap(corr , annot = True)
```

Out[17]:

```
<AxesSubplot:>
```

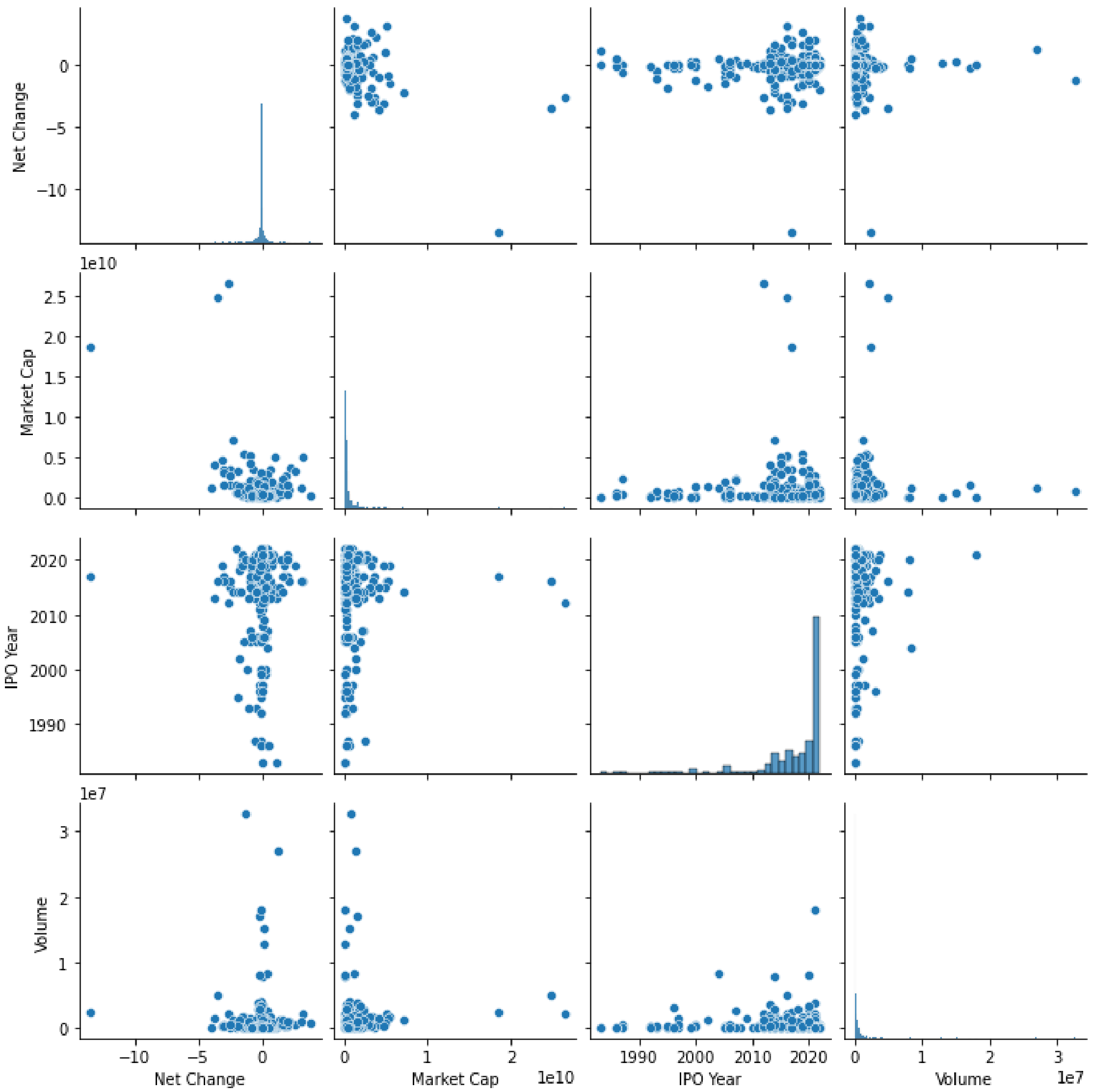


In [18]:

```
sns.pairplot(df)
```

Out[18]:

```
<seaborn.axisgrid.PairGrid at 0x1bc3a923d60>
```

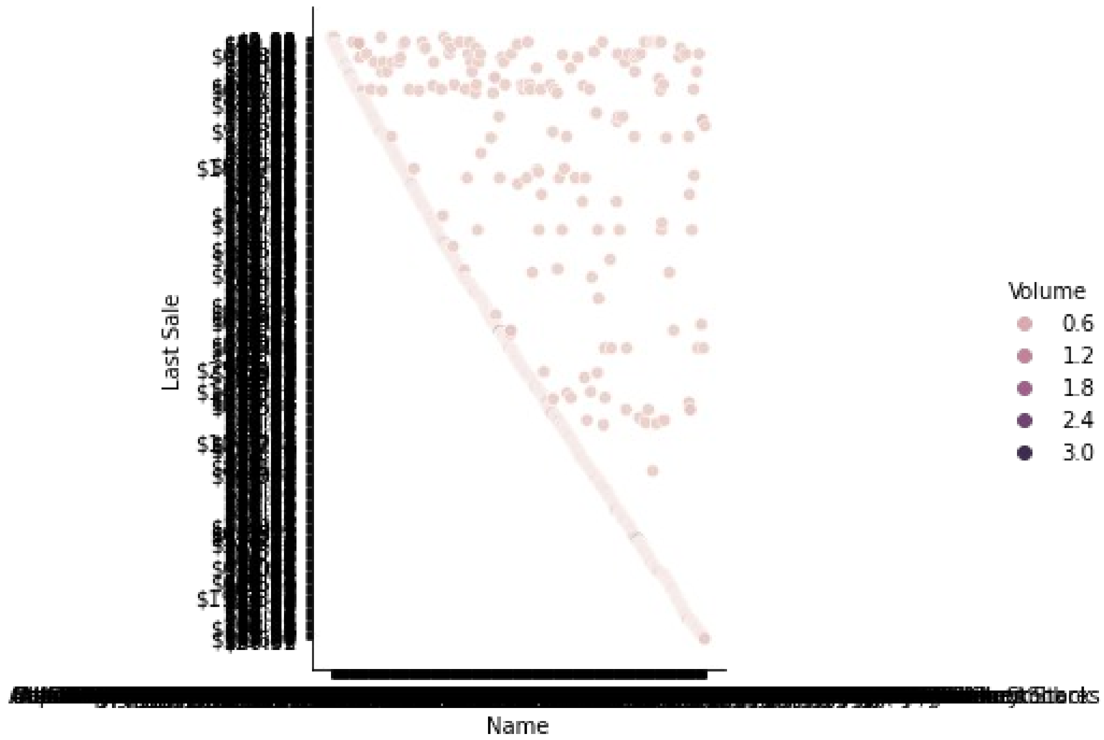


In [19]:

```
sns.relplot(x='Name', y='Last Sale', hue='Volume', data=df)
```

Out[19]:

```
<seaborn.axisgrid.FacetGrid at 0x1bc3beae3a0>
```



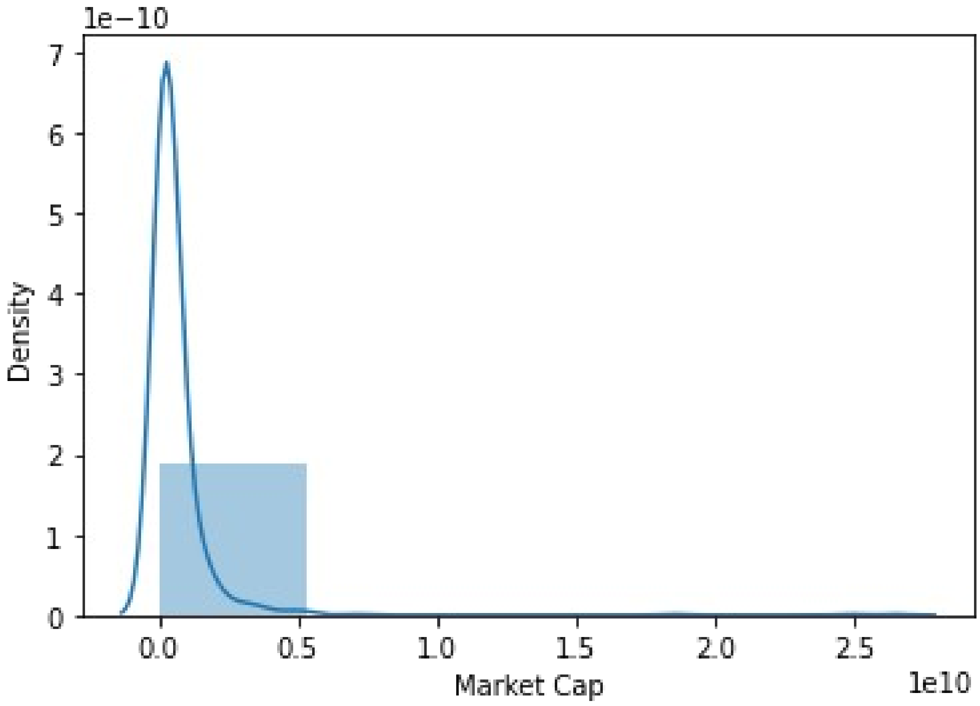
In [20]:

```
sns.distplot(df['Market Cap'] , bins=5)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\
distributions.py:2619: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[20]:

```
<AxesSubplot:xlabel='Market Cap', ylabel='Density'>
```

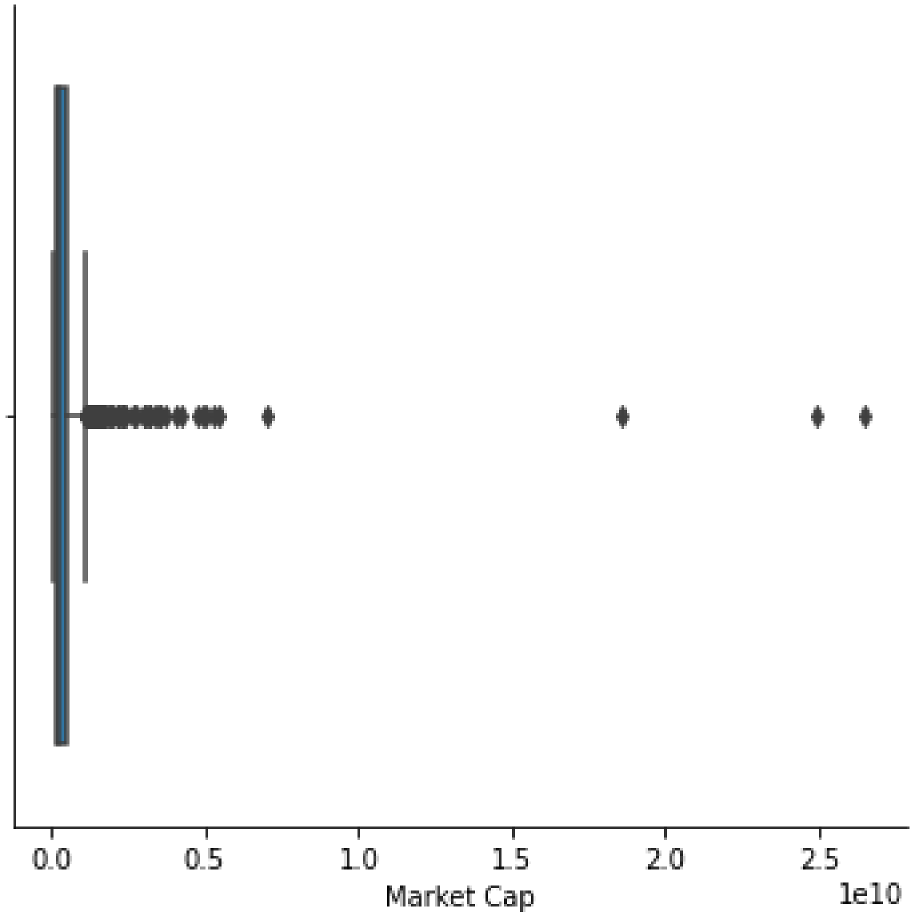


In [21]:

```
sns.catplot(x='Market Cap' , kind='box',  data=df)
```

Out[21]:

```
<seaborn.axisgrid.FacetGrid at 0x1bc3debd820>
```



In [22]:

```
numeric_data = df.select_dtypes(include=[np.number])
categorical_data = df.select_dtypes(exclude=[np.number])

print(f'No of Numerical feature present in numeric_data is
{numeric_data.shape[1]}')
print(f'No of Categorical feature present in categorical_data is
{categorical_data.shape[1]}')
```

```
No of Numerical feature present in numeric_data is 4
No of Categorical feature present in categorical_data is 7
```

In [23]:

```
categorical_data.columns
```

Out[23]:

```
Index(['Symbol', 'Name', 'Last Sale', '% Change', 'Country', 'Sector',
       'Industry'],
      dtype='object')
```

In [24]:

```
corr['Volume']
```

Out[24]:

```
Net Change   -0.026835
Market Cap    0.126539
IPO Year     -0.072968
Volume        1.000000
Name: Volume, dtype: float64
```

In [25]:

```
print("All the numrical columns are",end=' ')
print(numeric_data.columns)
```

```
All the numrical columns are Index(['Net Change', 'Market Cap', 'IPO
Year', 'Volume'], dtype='object')
```

In [26]:

```
print("All the categorical columns are",end=' ')
print(categorical_data.columns)
```

```
All the categorical columns are Index(['Symbol', 'Name', 'Last Sale',
'% Change', 'Country', 'Sector',
       'Industry'],
      dtype='object')
```

In [27]:

```
for col_name in categorical_data:
    print('*'*15 , col_name , '*'*15)
    print(df[col_name].value_counts().to_frame())
    print('*'*35 , end='\n')
```

```
*************** Symbol ***************
      Symbol
AACI       1
PCCT       1
ONYX       1
OPBK       1
OSTK       1
...      ...
FRBA       1
FRBK       1
FRG        1
FRPT       1
ZT         1

[678 rows x 1 columns]
***********************************
```

```
*************** Name ***************
                                                    Name
Armada Acquisition Corp. I Common Stock                1
Perception Capital Corp. II Class A Ordinary Sh...     1
Onyx Acquisition Co. I Class A Ordinary Shares         1
OP Bancorp Common Stock                                1
Overstock.com Inc. Common Stock                        1
...                                                  ...
First Bank Common Stock                                1
Republic First Bancorp Inc. Common Stock               1
Franchise Group Inc. Common Stock                      1
Freshpet Inc. Common Stock                             1
Zimmer Energy Transition Acquisition Corp. Clas...     1

[678 rows x 1 columns]
***********************************
*************** Last Sale ***************
        Last Sale
$9.98          15
$9.95          12
$9.93          11
$9.94          11
$9.97           9
...           ...
$8.19           1
$5.78           1
$1.25           1
$52.25          1
$27.02          1

[510 rows x 1 columns]
***********************************
*************** % Change ***************
        % Change
0.00%        134
0.10%         13
-0.20%        10
0.30%          8
0.20%          6
...          ...
-1.92%         1
-2.70%         1
-5.39%         1
0.96%          1
8.12%          1

[412 rows x 1 columns]
***********************************
*************** Country ***************
                      Country
```

```
United States             544
China                      29
Israel                     22
Canada                     10
Cayman Islands             10
United Kingdom             10
Germany                     5
Singapore                   5
Malaysia                    4
France                      4
Australia                   4
Mexico                      4
Switzerland                 3
Netherlands                 3
United Arab Emirates        3
Hong Kong                   3
Sweden                      2
Ireland                     2
Bermuda                     1
Belgium                     1
Taiwan                      1
South Korea                 1
Portugal                    1
Cyprus                      1
Jersey                      1
Macau                       1
Brazil                      1
Luxembourg                  1
***********************************
*************** Sector ***************
                        Sector
Health Care                187
Industrials                110
Finance                     79
Consumer Discretionary      79
Technology                  71
Miscellaneous                8
Consumer Staples             7
Real Estate                  6
Utilities                    6
Energy                       5
Telecommunications           3
***********************************
*************** Industry ***************
                                            Industry
Biotechnology: Pharmaceutical Preparations       125
Consumer Electronics/Appliances                   91
Major Banks                                       34
EDP Services                                      25
Specialty Chemicals                               19
```

```
...                                                ...
Cable & Other Pay Television Services                1
Natural Gas Distribution                             1
Specialty Insurers                                   1
Engineering & Construction                           1
Beverages (Production/Distribution)                  1

[93 rows x 1 columns]
***********************************
```

In [28]:

```
print(f"The datatype of Volume is {df.Volume.dtype}")
print(f"Datatype of df.Volume[0] is {type(df.Volume[0])} ")
```

```
The datatype of Volume is int64
Datatype of df.Volume[0] is <class 'numpy.int64'>
```

In [29]:

```
df.columns
```

Out[29]:

```
Index(['Symbol', 'Name', 'Last Sale', 'Net Change', '% Change',
'Market Cap',
       'Country', 'IPO Year', 'Volume', 'Sector', 'Industry'],
      dtype='object')
```

In [30]:

```
corr = df.corr()
pd.set_option("max_columns" , None)
corr[['Market Cap']].T
```

Out[30]:

Market Cap

In [31]:

```
corr = df.corr()
pd.set_option("max_columns" , None)
corr[['Volume']].T
```

Out[31]:

Volume

In [32]:

```
from sklearn.preprocessing import MinMaxScaler
```

In [33]:

```
df = pd.DataFrame(df , columns = df.columns )
```

In [34]:

```
df.head()
```

Out[34]:

	Symbol
0	AACI
1	AAME
2	AAOI
3	ABTX
4	ACAB

In [35]:

```
df.head(1)
```

Out[35]:

	Symbol
0	AACI

In [36]:

```
y = df.Symbol
x = df.drop(columns = ['Symbol'])
```

In [37]:

```
from sklearn.model_selection import train_test_split
x_train , x_test , y_train , y_test = train_test_split(x ,
y,train_size=0.8 , random_state =19)
```

In [38]:

```
y.head()
```

Out[38]:

```
0    AACI
1    AAME
2    AAOI
```

```
3    ABTX
4    ACAB
Name: Symbol, dtype: object
```

In [39]:

```
x.head()
```

Out[39]:

	Name
0	Armada Acquisition Corp. I Common Stock
1	Atlantic American Corporation Common Stock
2	Applied Optoelectronics Inc. Common Stock
3	Allegiance Bancshares Inc. Common Stock
4	Atlantic Coastal Acquisition Corp. II Class A ...

In [40]:

```
from sklearn.linear_model import LogisticRegression
```

In [41]:

```
model = LogisticRegression(fit_intercept=True, max_iter=10000)
```

In [43]:

```
model.fit(x_train , y_train)
```

```
---------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
~\AppData\Local\Temp/ipykernel_3432/1964293123.py in <module>
----> 1 model.fit(x_train , y_train)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\
_logistic.py in fit(self, X, y, sample_weight)
   1342             _dtype = [np.float64, np.float32]
   1343
-> 1344         X, y = self._validate_data(X, y, accept_sparse='csr',
dtype=_dtype,
   1345                                    order="C",
   1346
accept_large_sparse=solver != 'liblinear')

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py in
_validate_data(self, X, y, reset, validate_separately, **check_params)
    431                 y = check_array(y, **check_y_params)
    432             else:
--> 433                 X, y = check_X_y(X, y, **check_params)
    434             out = X, y
    435 

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py
in inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64 
     65             # extra_args > 0

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py
in check_X_y(X, y, accept_sparse, accept_large_sparse, dtype, order,
copy, force_all_finite, ensure_2d, allow_nd, multi_output,
ensure_min_samples, ensure_min_features, y_numeric, estimator)
    869         raise ValueError("y cannot be None")
    870 
--> 871     X = check_array(X, accept_sparse=accept_sparse,
    872                     accept_large_sparse=accept_large_sparse,
    873                     dtype=dtype, order=order, copy=copy,

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py
in inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64 
     65             # extra_args > 0

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py
in check_array(array, accept_sparse, accept_large_sparse, dtype,
order, copy, force_all_finite, ensure_2d, allow_nd,
ensure_min_samples, ensure_min_features, estimator)
    671                     array = array.astype(dtype,
casting="unsafe", copy=False)
    672                 else:
--> 673                     array = np.asarray(array, order=order,
dtype=dtype)
    674             except ComplexWarning as complex_warning:
    675                 raise ValueError("Complex data not supported\
n"

C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\_asarray.py in
asarray(a, dtype, order, like)
    100         return _asarray_with_like(a, dtype=dtype, order=order,
like=like)
    101 
--> 102     return array(a, dtype, copy=False, order=order)
```

```
    103
    104

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in
__array__(self, dtype)
   1991
   1992     def __array__(self, dtype: NpDtype | None = None) ->
np.ndarray:
-> 1993         return np.asarray(self._values, dtype=dtype)
   1994
   1995     def __array_wrap__(

C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\_asarray.py in
asarray(a, dtype, order, like)
    100         return _asarray_with_like(a, dtype=dtype, order=order,
like=like)
    101
--> 102     return array(a, dtype, copy=False, order=order)
    103
    104

ValueError: could not convert string to float: 'Puyi Inc. American
Depository Shares'
```

In [ ]:

```
y_pred = model.predict(x_test)
y_pred[0:5]
```

In [ ]:

In [ ]: