In [185…:
```python
import numpy as np
import pandas as pd
from datetime import datetime
```

In [56]:
```python
df = pd.read_csv('Orders.csv')
df
```

Out[56]:

	OrderID	InvoiceDate	CustomerID	OrderVolume	ShipMode	Segment	Category	PaymentTerm	Discount	ExistingPurchaseOrder	FirstCustomerOrder	CashInDate
0	O-100	2011-01-03	1024	276.1	Same Day	Corporate	Office Supplies	30	5%	Yes	Yes	2011-02-18
1	O-101	2011-01-03	1024	35.88	Same Day	Corporate	Office Supplies	30	5%	Yes	No	2011-02-18
2	O-102	2011-01-05	1006	66.12	Second Class	Consumer	Office Supplies	30	3%	Yes	Yes	2011-02-06
3	O-104	2011-01-06	1023	408.3	Standard Class	Consumer	Office Supplies	30	5%	No	Yes	2011-02-17
4	O-105	2011-01-06	1009	314.22	Standard Class	Consumer	Technology	45	3%	Yes	Yes	2011-02-10
...	...	...	...	...	...	...	...	...	...	...	...	...
37637	O-48951	2014/30/11	1029	50.09	Standard Class	Corporate	Office Supplies	14	3%	No	No	2014-12-31
37638	O-48953	2014/30/11	1010	21.12	Same Day	Consumer	Office Supplies	14	5%	Yes	No	2014-12-31
37639	O-48960	2014-11-30	1031	33.57	First Class	Consumer	Office Supplies	14	5%	Yes	No	2014-12-16
37640	O-48972	2014-11-30	1031	20.72	Second Class	Consumer	Office Supplies	30	3%	Yes	No	2014-12-15
37641	O-48977	2014-11-30	1000	1.2	Standard Class	Consumer	Office Supplies	14	0%	Yes	No	2014-12-30

37642 rows × 12 columns

In [57]:
```python
df['value'] = np.arange(len(df))
df
```

Out[57]:

	OrderID	InvoiceDate	CustomerID	OrderVolume	ShipMode	Segment	Category	PaymentTerm	Discount	ExistingPurchaseOrder	FirstCustomerOrder	CashInDate	value
0	O-100	2011-01-03	1024	276.1	Same Day	Corporate	Office Supplies	30	5%	Yes	Yes	2011-02-18	0
1	O-101	2011-01-03	1024	35.88	Same Day	Corporate	Office Supplies	30	5%	Yes	No	2011-02-18	1
2	O-102	2011-01-05	1006	66.12	Second Class	Consumer	Office Supplies	30	3%	Yes	Yes	2011-02-06	2
3	O-104	2011-01-06	1023	408.3	Standard Class	Consumer	Office Supplies	30	5%	No	Yes	2011-02-17	3
4	O-105	2011-01-06	1009	314.22	Standard Class	Consumer	Technology	45	3%	Yes	Yes	2011-02-10	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...
37637	O-48951	2014/30/11	1029	50.09	Standard Class	Corporate	Office Supplies	14	3%	No	No	2014-12-31	37637
37638	O-48953	2014/30/11	1010	21.12	Same Day	Consumer	Office Supplies	14	5%	Yes	No	2014-12-31	37638
37639	O-48960	2014-11-30	1031	33.57	First Class	Consumer	Office Supplies	14	5%	Yes	No	2014-12-16	37639
37640	O-48972	2014-11-30	1031	20.72	Second Class	Consumer	Office Supplies	30	3%	Yes	No	2014-12-15	37640
37641	O-48977	2014-11-30	1000	1.2	Standard Class	Consumer	Office Supplies	14	0%	Yes	No	2014-12-30	37641

37642 rows × 13 columns

In [58]:
```python
df.shape
```

Out[58]:
```
(37642, 13)
```

In [59]:
```python
df.isnull().sum()
```

Out[59]:
```
OrderID                   0
InvoiceDate               0
CustomerID                0
OrderVolume               9
ShipMode                  0
Segment                   0
Category                  0
PaymentTerm               0
Discount                 16
ExistingPurchaseOrder     0
FirstCustomerOrder        0
CashInDate                0
value                     0
dtype: int64
```

In [60]:
```python
df['OrderVolume'].value_counts()
```

Out[60]:
```
12.96      47
25.92      39
32.4       32
19.44      29
15.55      29
           ..
212.85      1
862.46      1
90.99       1
1007.74     1
421.28      1
Name: OrderVolume, Length: 16952, dtype: int64
```

In [61]:
```python
#Extracting column from dataframe
df_test = df[["InvoiceDate","CashInDate","value"]]
df_test
```

Out[61]:

	InvoiceDate	CashInDate	value
0	2011-01-03	2011-02-18	0
1	2011-01-03	2011-02-18	1
2	2011-01-05	2011-02-06	2
3	2011-01-06	2011-02-17	3
4	2011-01-06	2011-02-10	4
...	...	...	...
37637	2014/30/11	2014-12-31	37637
37638	2014/30/11	2014-12-31	37638
37639	2014-11-30	2014-12-16	37639
37640	2014-11-30	2014-12-15	37640
37641	2014-11-30	2014-12-30	37641

37642 rows × 3 columns

In [8]:
```python
#Coverting column to numeric format
df['OrderVolume'] = pd.to_numeric(df['OrderVolume'],errors = 'coerce')
df.OrderVolume.dtypes
```

Out[8]:
```
dtype('float64')
```

In [9]:
```python
fill = df['OrderVolume'].mean()
```

In [10]:
```python
#filling null values with mean
df['OrderVolume'] = df['OrderVolume'].fillna(df['OrderVolume'].mean())
```

In [11]:
```python
df.OrderVolume
```

Out[11]:
```
0        276.10
1         35.88
2         66.12
3        408.30
4        314.22
          ...
37637     50.09
37638     21.12
37639     33.57
37640     20.72
37641      1.20
Name: OrderVolume, Length: 37642, dtype: float64
```

In [15]:
```python
df_test.dtypes
```

Out[15]:
```
InvoiceDate    object
CashInDate     object
dtype: object
```

In [12]:
```python
df.loc[df['Discount'] == '5%','Discount'] = 5
df.loc[df['Discount'] == '3%','Discount'] = 3
df.loc[df['Discount'] == '0%','Discount'] = 0
```

In [13]:
```python
df.Discount.value_counts()
```

Out[13]:
```
5    18748
3     9440
0     9438
Name: Discount, dtype: int64
```

In [14]:
```python
mode_dis = df['Discount'].mode()
```

In [15]:
```python
mode_dis
```

Out[15]:
```
0    5
dtype: object
```

In [16]:
```python
mode_dis = 5
```

In [17]:
```python
df['Discount'] = df['Discount'].fillna(mode_dis)
```

In [19]:
```python
df = df.drop(['OrderID'],axis=1)
```

In [62]:
```python
df
```

Out[62]:

	OrderID	InvoiceDate	CustomerID	OrderVolume	ShipMode	Segment	Category	PaymentTerm	Discount	ExistingPurchaseOrder	FirstCustomerOrder	CashInDate	value
0	O-100	2011-01-03	1024	276.1	Same Day	Corporate	Office Supplies	30	5%	Yes	Yes	2011-02-18	0
1	O-101	2011-01-03	1024	35.88	Same Day	Corporate	Office Supplies	30	5%	Yes	No	2011-02-18	1
2	O-102	2011-01-05	1006	66.12	Second Class	Consumer	Office Supplies	30	3%	Yes	Yes	2011-02-06	2
3	O-104	2011-01-06	1023	408.3	Standard Class	Consumer	Office Supplies	30	5%	No	Yes	2011-02-17	3
4	O-105	2011-01-06	1009	314.22	Standard Class	Consumer	Technology	45	3%	Yes	Yes	2011-02-10	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...
37637	O-48951	2014/30/11	1029	50.09	Standard Class	Corporate	Office Supplies	14	3%	No	No	2014-12-31	37637
37638	O-48953	2014/30/11	1010	21.12	Same Day	Consumer	Office Supplies	14	5%	Yes	No	2014-12-31	37638
37639	O-48960	2014-11-30	1031	33.57	First Class	Consumer	Office Supplies	14	5%	Yes	No	2014-12-16	37639
37640	O-48972	2014-11-30	1031	20.72	Second Class	Consumer	Office Supplies	30	3%	Yes	No	2014-12-15	37640
37641	O-48977	2014-11-30	1000	1.2	Standard Class	Consumer	Office Supplies	14	0%	Yes	No	2014-12-30	37641

37642 rows × 13 columns

In [63]:
```python
#transforming dataframe to numeric
df.FirstCustomerOrder = df.FirstCustomerOrder.replace(['No','Yes'],[0,1])
df.ExistingPurchaseOrder = df.ExistingPurchaseOrder.replace(['No','Yes'],[0,1])
```

In [64]:
```python
from sklearn.preprocessing import LabelEncoder
```

In [65]:
```python
#encoding few colmns for future needs
label = LabelEncoder()
df.ShipMode = label.fit_transform(df['ShipMode'])
df.Segment = label.fit_transform(df['Segment'])
df.Category = label.fit_transform(df['Category'])
```

In [55]:
```python
df
```

Out[55]:

	InvoiceDate	CustomerID	OrderVolume	ShipMode	Segment	Category	PaymentTerm	Discount	ExistingPurchaseOrder	FirstCustomerOrder	CashInDate	value
0	2011-01-03	1024	276.10	1	1	1	30	5	1	1	2011-02-18	0
1	2011-01-03	1024	35.88	1	1	1	30	5	1	0	2011-02-18	1
2	2011-01-05	1006	66.12	2	0	1	30	3	1	1	2011-02-06	2
3	2011-01-06	1023	408.30	3	0	1	30	5	0	1	2011-02-17	3
4	2011-01-06	1009	314.22	3	0	2	45	3	1	1	2011-02-10	4
...	...	...	...	...	...	...	...	...	...	...	...	...
37637	2014/30/11	1029	50.09	3	1	1	14	3	0	0	2014-12-31	37637
37638	2014/30/11	1010	21.12	1	0	1	14	5	1	0	2014-12-31	37638
37639	2014-11-30	1031	33.57	0	0	1	14	5	1	0	2014-12-16	37639
37640	2014-11-30	1031	20.72	2	0	1	30	3	1	0	2014-12-15	37640
37641	2014-11-30	1000	1.20	3	0	1	14	0	1	0	2014-12-30	37641

37642 rows × 12 columns

In [66]:
```python
#converting into datetime format
df_test = df[['InvoiceDate','CashInDate']].apply(lambda _: pd.to_datetime(_,format='%Y-%m-%d %H:%M:%S.%f', errors='coerce'))
df_test
```

Out[66]:

	InvoiceDate	CashInDate
0	2011-01-03	2011-02-18
1	2011-01-03	2011-02-18
2	2011-01-05	2011-02-06
3	2011-01-06	2011-02-17
4	2011-01-06	2011-02-10
...	...	...
37637	NaT	2014-12-31
37638	NaT	2014-12-31
37639	2014-11-30	2014-12-16
37640	2014-11-30	2014-12-15
37641	2014-11-30	2014-12-30

37642 rows × 2 columns

In [26]:
```python
df_test.dtypes
```

Out[26]:
```
InvoiceDate    datetime64[ns]
CashInDate     datetime64[ns]
dtype: object
```

In [67]:
```python
df_test['Difference'] = df_test['CashInDate'].sub(df_test['InvoiceDate'], axis=0)
df_test['Difference'] = df_test['Difference']/np.timedelta64(1,'D')
```

In [68]:
```python
df_test
```

Out[68]:

	InvoiceDate	CashInDate	Difference
0	2011-01-03	2011-02-18	46.0
1	2011-01-03	2011-02-18	46.0
2	2011-01-05	2011-02-06	32.0
3	2011-01-06	2011-02-17	42.0
4	2011-01-06	2011-02-10	35.0
...	...	...	...
37637	NaT	2014-12-31	NaN
37638	NaT	2014-12-31	NaN
37639	2014-11-30	2014-12-16	16.0
37640	2014-11-30	2014-12-15	15.0
37641	2014-11-30	2014-12-30	30.0

37642 rows × 3 columns

In [98]:
```python
df_test.dropna(axis=0 , inplace=True)
df_test
```

Out[98]:

	Difference
0	46.0
1	46.0
2	32.0
3	42.0
4	35.0
...	...
37631	29.0
37632	22.0
37639	16.0
37640	15.0
37641	30.0

37628 rows × 1 columns

In [99]:
```python
result = pd.concat([df, df_test], axis=1, join='inner')
display(result)
```

	OrderID	InvoiceDate	CustomerID	OrderVolume	ShipMode	Segment	Category	PaymentTerm	Discount	ExistingPurchaseOrder	FirstCustomerOrder	CashInDate	value	Difference
0	O-100	2011-01-03	1024	276.1	1	1	1	30	5%	1	1	2011-02-18	0	46.0
1	O-101	2011-01-03	1024	35.88	1	1	1	30	5%	1	0	2011-02-18	1	46.0
2	O-102	2011-01-05	1006	66.12	2	0	1	30	3%	1	1	2011-02-06	2	32.0
3	O-104	2011-01-06	1023	408.3	3	0	1	30	5%	0	1	2011-02-17	3	42.0
4	O-105	2011-01-06	1009	314.22	3	0	2	45	3%	1	1	2011-02-10	4	35.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
37631	O-48922	2014-11-30	1013	79.9	0	0	1	14	5%	1	0	2014-12-29	37631	29.0
37632	O-48929	2014-11-30	1020	151.88	3	0	1	14	3%	1	0	2014-12-22	37632	22.0
37639	O-48960	2014-11-30	1031	33.57	0	0	1	14	5%	1	0	2014-12-16	37639	16.0
37640	O-48972	2014-11-30	1031	20.72	2	0	1	30	3%	1	0	2014-12-15	37640	15.0
37641	O-48977	2014-11-30	1000	1.2	3	0	1	14	0%	1	0	2014-12-30	37641	30.0

37628 rows × 14 columns

In [102…:
```python
data_final = result[["InvoiceDate","value","Difference"]]
data_final
```

Out[102…:

	InvoiceDate	value	Difference
0	2011-01-03	0	46.0
1	2011-01-03	1	46.0
2	2011-01-05	2	32.0
3	2011-01-06	3	42.0
4	2011-01-06	4	35.0
...	...	...	...
37631	2014-11-30	37631	29.0
37632	2014-11-30	37632	22.0
37639	2014-11-30	37639	16.0
37640	2014-11-30	37640	15.0
37641	2014-11-30	37641	30.0

37628 rows × 3 columns

In [103…:
```python
data_final.isnull().sum()
```

Out[103…:
```
InvoiceDate    0
value          0
Difference     0
dtype: int64
```

In [104…:
```python
df_index = data_final.set_index('InvoiceDate')
df_index
```

Out[104…:

InvoiceDate	value	Difference
2011-01-03	0	46.0
2011-01-03	1	46.0
2011-01-05	2	32.0
2011-01-06	3	42.0
2011-01-06	4	35.0
...	...	...
2014-11-30	37631	29.0
2014-11-30	37632	22.0
2014-11-30	37639	16.0
2014-11-30	37640	15.0
2014-11-30	37641	30.0

37628 rows × 2 columns

In [116…:
```python
df_index['Difference1'] = df_index['Difference'].shift(-1)
df_index['Difference2'] = df_index['Difference'].shift(-2)
df_index
```

Out[116…:

InvoiceDate	value	Difference	Difference1	Difference2
2011-01-03	0	46.0	46.0	32.0
2011-01-03	1	46.0	32.0	42.0
2011-01-05	2	32.0	42.0	35.0
2011-01-06	3	42.0	35.0	33.0
2011-01-06	4	35.0	33.0	40.0
...	...	...	...	...
2014-11-30	37631	29.0	22.0	16.0
2014-11-30	37632	22.0	16.0	15.0
2014-11-30	37639	16.0	15.0	30.0
2014-11-30	37640	15.0	30.0	NaN
2014-11-30	37641	30.0	NaN	NaN

37628 rows × 4 columns

In [106…:
```python
df_index.dropna
df_index
```

Out[106…:

InvoiceDate	value	Difference	Difference1	Difference2
2011-01-03	0	46.0	46.0	32.0
2011-01-03	1	46.0	32.0	42.0
2011-01-05	2	32.0	42.0	35.0
2011-01-06	3	42.0	35.0	33.0
2011-01-06	4	35.0	33.0	40.0
...	...	...	...	...
2014-11-30	37631	29.0	22.0	16.0
2014-11-30	37632	22.0	16.0	15.0
2014-11-30	37639	16.0	15.0	30.0
2014-11-30	37640	15.0	30.0	NaN
2014-11-30	37641	30.0	NaN	NaN

37628 rows × 4 columns

In [123…:
```python
df_index.dropna
```

Out[123…:
```
<bound method DataFrame.dropna of              value  Difference  Difference1  Difference2
InvoiceDate
2011-01-03       0        46.0         46.0         32.0
2011-01-03       1        46.0         32.0         42.0
2011-01-05       2        32.0         42.0         35.0
2011-01-06       3        42.0         35.0         33.0
2011-01-06       4        35.0         33.0         40.0
...            ...         ...          ...          ...
2014-11-30   37631        29.0         22.0         16.0
2014-11-30   37632        22.0         16.0         15.0
2014-11-30   37639        16.0         15.0         30.0
2014-11-30   37640        15.0         30.0          NaN
2014-11-30   37641        30.0          NaN          NaN

[37628 rows x 4 columns]>
```

In [124…:
```python
df_index.isnull().sum()
```

Out[124…:
```
value          0
Difference     0
Difference1    1
Difference2    2
dtype: int64
```

In [107…:
```python
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
```

In [108…:
```python
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators=100,max_features = 3,random_state =1)
```

In [117…:
```python
import numpy as np
x,y = df_index['value'],df_index['Difference']
x,y = np.array(x),np.array(y)
x,y= x.reshape(-1,1), y.reshape(-1,1)
print(x)
```

```
[[    0]
 [    1]
 [    2]
 ...
 [37639]
 [37640]
 [37641]]
```

In [118…:
```python
print(y)
```

```
[[46.]
 [46.]
 [32.]
 ...
 [16.]
 [15.]
 [30.]]
```

In [125…:
```python
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size = 20,random_state = 1)
```

In [126…:
```python
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(xtrain,ytrain)
```

Out[126…:
```
LinearRegression()
```

In [127…:
```python
reg.fit(xtrain,ytrain)
```

Out[127…:
```
LinearRegression()
```

In [133…:
```python
pred = reg.predict(xtest)
pred
```

Out[133…:
```
array([[35.68473249],
       [35.68572946],
       [35.68414179],
       [35.68440241],
       [35.68410374],
       [35.68629835],
       [35.68497606],
       [35.68437811],
       [35.68585514],
       [35.68494916],
       [35.68610607],
       [35.6846081 ],
       [35.68643519],
       [35.68653034],
       [35.68354142],
       [35.68623538],
       [35.68422725],
       [35.68493402],
       [35.68394804],
       [35.68493783]])
```

In [138…:
```python
reg.predict([[1], [7], [50]])
```

Out[138…:
```
array([[35.68653518],
       [35.68653466],
       [35.68653094]])
```

In [139…:
```python
from sklearn.metrics import mean_squared_error
from math import sqrt
rmse_lr = sqrt(mean_squared_error(pred,ytest))
```

In [141…:
```python
rmse_lr = ('Mean Squared Error for Linear Regression Model is :', rmse_lr)

rmse_lr
```

Out[141…:
```
('Mean Squared Error for Linear Regression Model is :',
 ('Mean Squared Error for Linear Regression Model is :', 13.344037564428815))
```

In [168…:
```python
reg.score(xtest,ytest)
```

Out[168…:
```
-0.008000784154470919
```

In [143…:
```python
num = 20
trange = pd.date_range('2014-12-1', periods=num, freq='d')
trange
```

Out[143…:
```
DatetimeIndex(['2014-12-01', '2014-12-02', '2014-12-03', '2014-12-04',
               '2014-12-05', '2014-12-06', '2014-12-07', '2014-12-08',
               '2014-12-09', '2014-12-10', '2014-12-11', '2014-12-12',
               '2014-12-13', '2014-12-14', '2014-12-15', '2014-12-16',
               '2014-12-17', '2014-12-18', '2014-12-19', '2014-12-20'],
              dtype='datetime64[ns]', freq='D')
```

In [142…:
```python
Forecast = reg.predict(xtest)
print(Forecast)
```

```
[[35.68473249]
 [35.68572946]
 [35.68414179]
 [35.68440241]
 [35.68410374]
 [35.68629835]
 [35.68497606]
 [35.68437811]
 [35.68585514]
 [35.68494916]
 [35.68610607]
 [35.6846081 ]
 [35.68643519]
 [35.68653034]
 [35.68354142]
 [35.68623538]
 [35.68422725]
 [35.68493402]
 [35.68394804]
 [35.68493783]]
```

In [144…:
```python
Predict_df = pd.DataFrame(Forecast, index=trange)
Predict_df.columns = ['forecast']
Predict_df
```

Out[144…:

	forecast
2014-12-01	35.684732
2014-12-02	35.685729
2014-12-03	35.684142
2014-12-04	35.684402
2014-12-05	35.684104
2014-12-06	35.686298
2014-12-07	35.684976
2014-12-08	35.684378
2014-12-09	35.685855
2014-12-10	35.684949
2014-12-11	35.686106
2014-12-12	35.684608
2014-12-13	35.686435
2014-12-14	35.686530
2014-12-15	35.683541
2014-12-16	35.686235
2014-12-17	35.684227
2014-12-18	35.684934
2014-12-19	35.683948
2014-12-20	35.684938

In [159…:
```python
from sklearn import ensemble
```

In [154…:
```python
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size = 20,random_state = 1)
```

In [160…:
```python
from sklearn.linear_model import LogisticRegression
```

In [164…:
```python
model = LogisticRegression()
```

In [165…:
```python
model.fit(xtrain,ytrain)
```

```
C:\Users\shahidul Alam\anaconda3\envs\Data_Science\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
C:\Users\shahidul Alam\anaconda3\envs\Data_Science\lib\site-packages\sklearn\linear_model\_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[165…:
```
LogisticRegression()
```

In [166…:
```python
model.predict(xtest)
```

Out[166…:
```
array([36., 36., 36., 36., 36., 36., 36., 36., 36., 36., 36., 36., 36.,
       36., 36., 36., 36., 36., 36., 36.])
```

In [169…:
```python
model.score(xtest,ytest)
```

Out[169…:
```
0.0
```

In [170…:
```python
from sklearn.ensemble import RandomForestClassifier
```

In [171…:
```python
Rclf = RandomForestClassifier()
```

In [172…:
```python
Rclf.fit(xtrain,ytrain)
```

```
<ipython-input-172-cc60e83c4be4>:1: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  Rclf.fit(xtrain,ytrain)
```

Out[172…:
```
RandomForestClassifier()
```

In [174…:
```python
Rclf.predict(xtest)
```

Out[174…:
```
array([42., 44., 50., 36., 60., 38., 33., 52., 52., 34., 39., 28., 25.,
       28.,  6., 46., 61., 41., 28., 38.])
```

In [176…:
```python
Rclf.score(xtest,ytest)
```

Out[176…:
```
0.0
```

In [184…:
```python
reg.predict(xtest)
```

Out[184…:
```
array([[35.68473249],
       [35.68572946],
       [35.68414179],
       [35.68440241],
       [35.68410374],
       [35.68629835],
       [35.68497606],
       [35.68437811],
       [35.68585514],
       [35.68494916],
       [35.68610607],
       [35.6846081 ],
       [35.68643519],
       [35.68653034],
       [35.68354142],
       [35.68623538],
       [35.68422725],
       [35.68493402],
       [35.68394804],
       [35.68493783]])
```

In [ ]: