NON-FUNGIBLE TOKEN (NFG) WITH MACHINE LEARNING

SUBMITTED BY

ASHWIN M NAYAK

REKHA J E

SHILPA S

ashwinnayak268@gmail.com

tocontactrekha@gmail.com

shillpashreeramappa@gmail.com

https://github.com/smartinternz02/SBSPS-Challenge-9524-NFT-Sales-Analytics-Dashboard/tree/main

OCTOBER 16, 2022

Smart Internz Project Report Titles

INTRODUCTION:

Non-fungible tokens (NFTs) are cryptographic assets on a blockchain with unique identification codes and metadata that distinguish them from each other. Unlike cryptocurrencies, they cannot be traded or exchanged at equivalency. This differs from fungible tokens like cryptocurrencies, which are identical to each other and, therefore, can serve as a medium for commercial transactions.

To offer a tangible solution for this problem, we are making use of machine learning and the relevant dataset used for the analysis are taken from the below link https://www.kaggle.com/datasets/mathurinache/nft-history-sales

2. LITERATURE SURVEY Existing problem:

NFT buyers and sellers are experiencing problems while visualizing and analysing the NFT sales transactions.

Solution:

We are providing a dashboard on NFT sales that helps the buyers and sellers for the better understanding of all the NFT transactions and also to learn more about NFT analytics dashboards and reports can be made. Traders, builders, and collectors might get a competitive edge in a brand-new market by applying analytics tools to spot patterns and anomalies. We will visualize using graph for better understanding.

3. HARDWARE/SOFTWARE ANALYSIS

Hardware:

RAM	**Minimum 4GB**
Processor	**Intel i3/Amd Ryzen 3**
HDD	**Minimum 250GB**

Software:

OS	**Windows/Linux/Mac**
Language	**Python/Python Library**

4. EXPERIMENTAL INVESTIGATIONS

The following summarises the overall strategy used in this project:

- **Data Preparation**: To create a useful dataset for modelling, a significant amount of work was spent on data preparation.
- **Data Pre-processing**: Clean the data by filling in missing values, smoothing the noisy data, resolving the inconsistency and removing the outliers.
- **Machine Learning Model Implementation**: Implementing Linear Regression and Random Forest to generate the accuracy of the predictions.
- **Creating a Dashboard**: We are using Cognos Analytics Dashboard and machine learning Algorithms to visualize data.
- **Language Used**: Python

5. FUTURE SCOPE

Right now, the NFT market is booming. According to **NFTGO** data, since May 2021, at least one NFT project has been released onto the chain every single day. Because NFT projects' quality can vary, just like DeFi's does, investors may be susceptible to the "liquidity trap".

6. RESULT

The designed algorithm can handle the inputs given by the user and based on the new inputs the algorithm is capable of producing the near futuristic values predictions, from which the users can foresee the predictions and act accordingly

7. APPLICATIONS

Right now, the NFT market is booming. According to **NFTGO** data, since May 2021, at least one NFT project has been released onto the chain every single day. Because NFT projects' quality can vary, just like DeFi's does, investors may be susceptible to the "liquidity trap".

8. CONCLUSION

Through these machine learning algorithms and visualization techniques, the buyers can get a better understanding of which NFT is beneficial and the seller can analyze and track the NFT sales transactions. That helps to increase their business and the seller can upscale their business.

9. BIBILOGRAPHY

https://youtu.be/nouFQezQm1A

https://us1.ca.analytics.ibm.com/bi/?perspective=dashboard&pathRef=.my_folders%2FNew%2Bdashboard&action=view&mode=dashboard&subView=model00000183e1af6577_00000003

10. Append Code

```
import pandas as pd
import numpy as np
from sklearn import linear_model
import seaborn as sns
```

```
pwd
```

```
'/home/wsuser/work'
```

```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
client_11e2c12b890445dc88de6b1cc18e27d0 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='8LroDiIvacCkB21uoNYN0Up03JuE0oaM7x8y7badPPrC',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

body = client_11e2c12b890445dc88de6b1cc18e27d0.get_object(Bucket='nftsalesanalytics-donotdelete-pr-hapwycfiwmwvwu',Key='NFT_Sales.csv')['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

df= pd.read_csv(body)
df.head()
```

Ln 13, Col 22 (17 selected)

```python
df= pd.read_csv(body)
df.head()
```

	Date	Sales_USD_cumsum	Number_of_Sales_cumsum	Active_Market_Wallets_cumsum	Primary_Sales_cumsum	Secondary_Sales_cumsum	AverageUSD_cum	Sales_USD	Number_of_Sales	Active_Mark
0	2017-06-22	0.00	0	0.0	0	NaN	NaN	NaN	NaN	
1	2017-06-23	1020.30	19	8.0	0	19.0	53.70	1020.30	19.0	
2	2017-06-24	2261.14	40	21.0	0	21.0	56.53	1240.84	21.0	
3	2017-06-25	2778.69	53	28.0	0	13.0	52.43	517.55	13.0	
4	2017-06-26	3203.32	67	34.0	0	14.0	47.81	424.63	14.0	

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

```
Date                            0
Sales_USD_cumsum                0
Number_of_Sales_cumsum          0
Active_Market_Wallets_cumsum    8
Primary_Sales_cumsum            0
Secondary_Sales_cumsum          1
AverageUSD_cum                  1
Sales_USD                       1
Number_of_Sales                 1
Active_Market_Wallets           9
Primary_Sales                   1
dtype: int64
```

```python
df['Active_Market_Wallets_cumsum'] = df['Active_Market_Wallets_cumsum'].fillna(df['Active_Market_Wallets_cumsum'].mean())
df['Active_Market_Wallets'] = df['Active_Market_Wallets'].fillna(df['Active_Market_Wallets'].mean())
df = df.fillna(0)
```

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

```
Date                            0
Sales_USD_cumsum                0
Number_of_Sales_cumsum          0
Active_Market_Wallets_cumsum    0
Primary_Sales_cumsum            0
Secondary_Sales_cumsum          0
AverageUSD_cum                  0
Sales_USD                       0
Number_of_Sales                 0
Active_Market_Wallets           0
Primary_Sales                   0
dtype: int64
```

```python
df.head(10)
```

	Date	Sales_USD_cumsum	Number_of_Sales_cumsum	Active_Market_Wallets_cumsum	Primary_Sales_cumsum	Secondary_Sales_cumsum	AverageUSD_cum	Sales_USD	Number_of_Sales	Active_Mar
0	2017-06-22	0.00	0	0.0	0	0.0	0.00	0.00	0.0	
1	2017-06-23	1020.30	19	8.0	0	19.0	53.70	1020.30	19.0	
2	2017-06-24	2261.14	40	21.0	0	21.0	56.53	1240.84	21.0	
3	2017-06-25	2778.69	53	28.0	0	13.0	52.43	517.55	13.0	
4	2017-06-26	3203.32	67	34.0	0	14.0	47.81	424.63	14.0	
5	2017-06-27	5296.23	100	44.0	0	33.0	52.96	2092.91	33.0	
6	2017-06-28	6543.49	115	46.0	0	15.0	56.90	1247.26	15.0	
7	2017-06-29	8729.21	153	49.0	0	38.0	57.05	2185.72	38.0	
8	2017-06-30	10437.36	174	53.0	0	21.0	59.98	1708.15	21.0	
9	2017-07-01	13578.16	184	55.0	0	10.0	73.79	3140.80	10.0	

```
df1=df.iloc[:,[9,10,5,6]]
df1
```

	Active_Market_Wallets	Primary_Sales	Secondary_Sales_cumsum	AverageUSD_cum
0	502.875391	0.0	0.0	0.00
1	8.000000	0.0	19.0	53.70
2	13.000000	0.0	21.0	56.53
3	7.000000	0.0	13.0	52.43
4	6.000000	0.0	14.0	47.81
...	...	...	...	...
1601	502.875391	44435.0	14829.0	924.39
1602	502.875391	32156.0	18723.0	924.84
1603	502.875391	27694.0	17128.0	926.44
1604	502.875391	7808.0	6127.0	928.49
1605	502.875391	-1171.0	-1024.0	928.11

1606 rows × 4 columns


```
df1=df.iloc[:,[9,10,5,6]]
df1
```

	Active_Market_Wallets	Primary_Sales	Secondary_Sales_cumsum	AverageUSD_cum
0	502.875391	0.0	0.0	0.00
1	8.000000	0.0	19.0	53.70
2	13.000000	0.0	21.0	56.53
3	7.000000	0.0	13.0	52.43
4	6.000000	0.0	14.0	47.81
...	...	...	...	...
1601	502.875391	44435.0	14829.0	924.39
1602	502.875391	32156.0	18723.0	924.84
1603	502.875391	27694.0	17128.0	926.44
1604	502.875391	7808.0	6127.0	928.49
1605	502.875391	-1171.0	-1024.0	928.11

1606 rows × 4 columns

```
x=df.drop(['AverageUSD_cum','Date','Sales_USD_cumsum'],axis=1)
y=df['AverageUSD_cum']
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)
```

```
x_train.shape,x_test.shape
```

```
((1076, 8), (530, 8))
```

```
pip install category_encoders
```

```
Collecting category_encoders
  Downloading category_encoders-2.5.1.post0-py2.py3-none-any.whl (72 kB)
     |████████████████████████████████| 72 kB 1.7 MB/s  eta 0:00:01
Requirement already satisfied: patsy>=0.5.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from category_encoders) (0.5.2)
Requirement already satisfied: numpy>=1.14.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from category_encoders) (1.20.3)
Requirement already satisfied: statsmodels>=0.9.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from category_encoders) (0.12.2)
Requirement already satisfied: pandas>=1.0.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from category_encoders) (1.3.4)
Requirement already satisfied: scikit-learn>=0.20.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from category_encoders) (1.0.2)
Requirement already satisfied: scipy>=1.0.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from category_encoders) (1.7.3)
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas>=1.0.5->category_encoders) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas>=1.0.5->category_encoders) (2021.3)
Requirement already satisfied: six in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from patsy>=0.5.1->category_encoders) (1.15.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from scikit-learn>=0.20.0->category_encoders) (2.2.0)
Requirement already satisfied: joblib>=0.11 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from scikit-learn>=0.20.0->category_encoders) (0.17.0)
```

Ln 14, Col 46   Go Live

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)
```

Pytho

```
x_train.shape,x_test.shape
```

Pytho

```
((1076, 8), (530, 8))
```

```
pip install category_encoders
```

Pytho

```
Collecting category_encoders
  Downloading category_encoders-2.5.1.post0-py2.py3-none-any.whl (72 kB)
     |████████████████████████████████| 72 kB 1.7 MB/s  eta 0:00:01
Requirement already satisfied: patsy>=0.5.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from category_encoders) (0.5.2)
Requirement already satisfied: numpy>=1.14.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from category_encoders) (1.20.3)
Requirement already satisfied: statsmodels>=0.9.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from category_encoders) (0.12.2)
Requirement already satisfied: pandas>=1.0.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from category_encoders) (1.3.4)
Requirement already satisfied: scikit-learn>=0.20.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from category_encoders) (1.0.2)
Requirement already satisfied: scipy>=1.0.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from category_encoders) (1.7.3)
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas>=1.0.5->category_encoders) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas>=1.0.5->category_encoders) (2021.3)
Requirement already satisfied: six in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from patsy>=0.5.1->category_encoders) (1.15.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from scikit-learn>=0.20.0->category_encoders) (2.2.0)
Requirement already satisfied: joblib>=0.11 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from scikit-learn>=0.20.0->category_encoders) (0.17.0)
Installing collected packages: category-encoders
Successfully installed category-encoders-2.5.1.post0
Note: you may need to restart the kernel to use updated packages.
```

```python
import category_encoders as ce
encoder=ce.OrdinalEncoder(cols=['Active_Market_Wallets','Primary_Sales','Secondary_Sales_cumsum'])
x_train=encoder.fit_transform(x_train)
x_test=encoder.transform(x_test)
```

```python
x_train.head()
```

	Number_of_Sales_cumsum	Active_Market_Wallets_cumsum	Primary_Sales_cumsum	Secondary_Sales_cumsum	Sales_USD	Number_of_Sales	Active_Market_Wallets	Primary_Sales
829	3246910	123838.0	2325117	1	74120.41	2563.0	1	1
126	958	141.0	0	2	268.92	7.0	2	2
916	3762801	135962.0	2570905	3	64775.42	5509.0	3	3
1187	4886901	177483.0	3127103	4	429816.64	2323.0	4	4
355	1339986	78339.0	885417	5	36981.38	2903.0	5	5

```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import *
```

```python
x_train,y_train = make_regression(n_features=4, n_informative=2, random_state=0,shuffle=False)
model = RandomForestRegressor(max_depth=2,random_state=0)
```

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

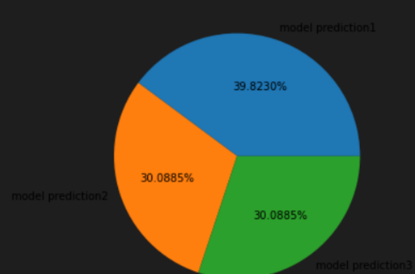
```
RandomForestRegressor(max_depth=2, random_state=0)
```

Ln 14, Col 46 Go Liv

```python
val1=int(model.predict([[1,1,1,1]]))
val2=int(model.predict([[0,1,0,1]]))
val3=int(model.predict([[0,1,1,1]]))
print(val1,val2,val3)
```

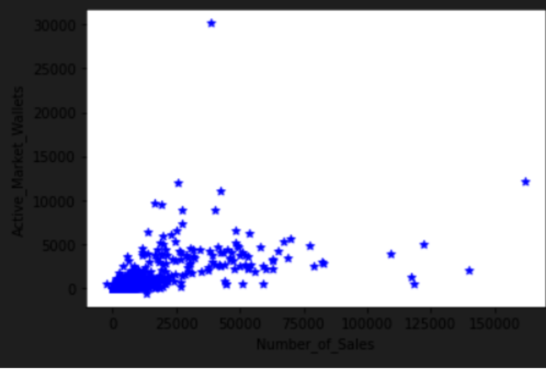
```
45 34 34
```

```python
import matplotlib.pyplot as plt
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.axis('equal')
l=['model prediction1','model prediction2','model prediction3']
s=[val1,val2,val3]
ax.pie(s,labels=l,autopct='%1.4f%%')
plt.show()
```



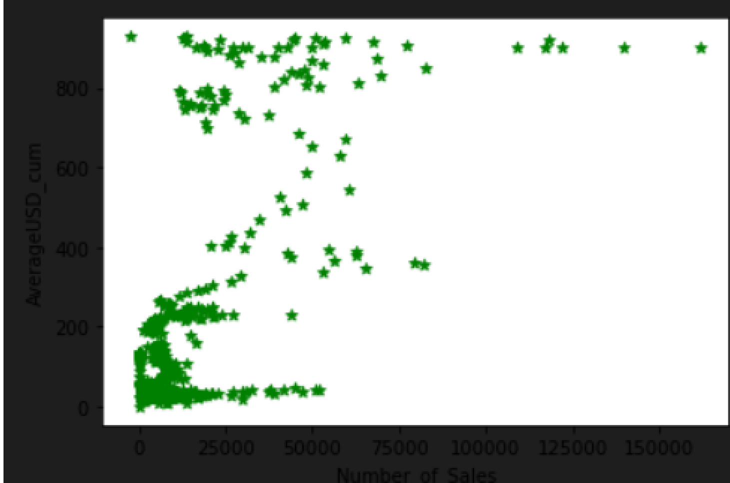
```
plt.xlabel('Number_of_Sales')
plt.ylabel('Active_Market_Wallets')
plt.scatter(df.Number_of_Sales,df.Active_Market_Wallets,color='blue',marker='*')
```

```
<matplotlib.collections.PathCollection at 0x7fd80fca4e80>
```

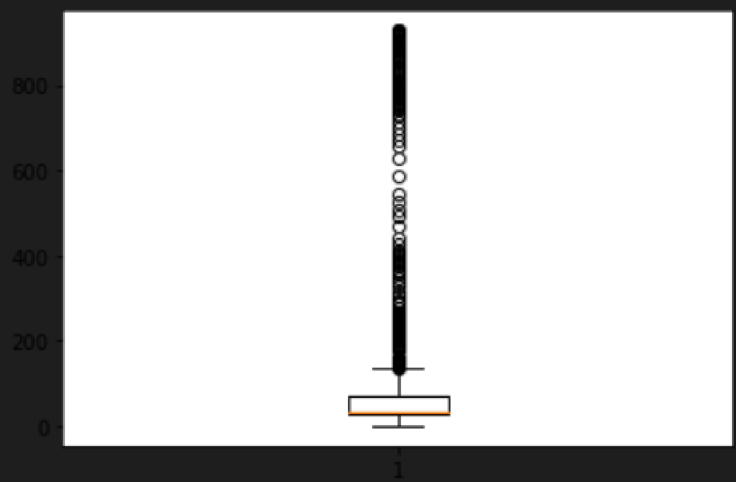



```
plt.xlabel('Number_of_Sales')
plt.ylabel('AverageUSD_cum')
plt.scatter(df.Number_of_Sales,df.AverageUSD_cum,color='green',marker='*')
```

```
<matplotlib.collections.PathCollection at 0x7fd807c3b220>
```

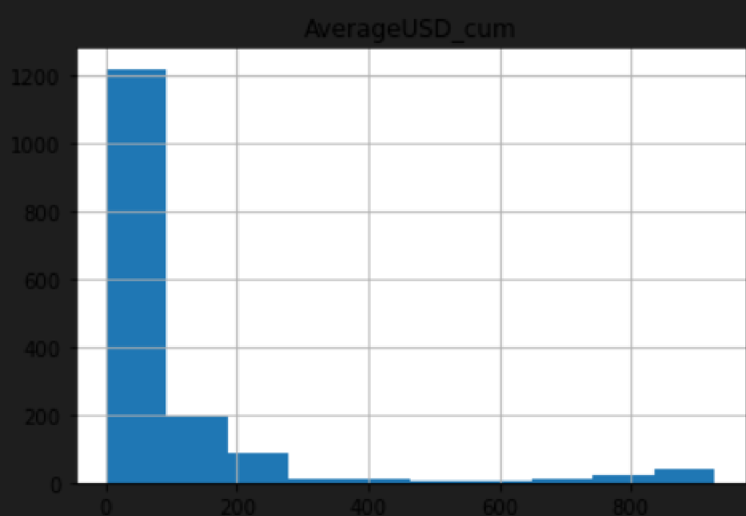


```
plt.figure(1)
plt.boxplot([df['AverageUSD_cum']])
plt.show()
```

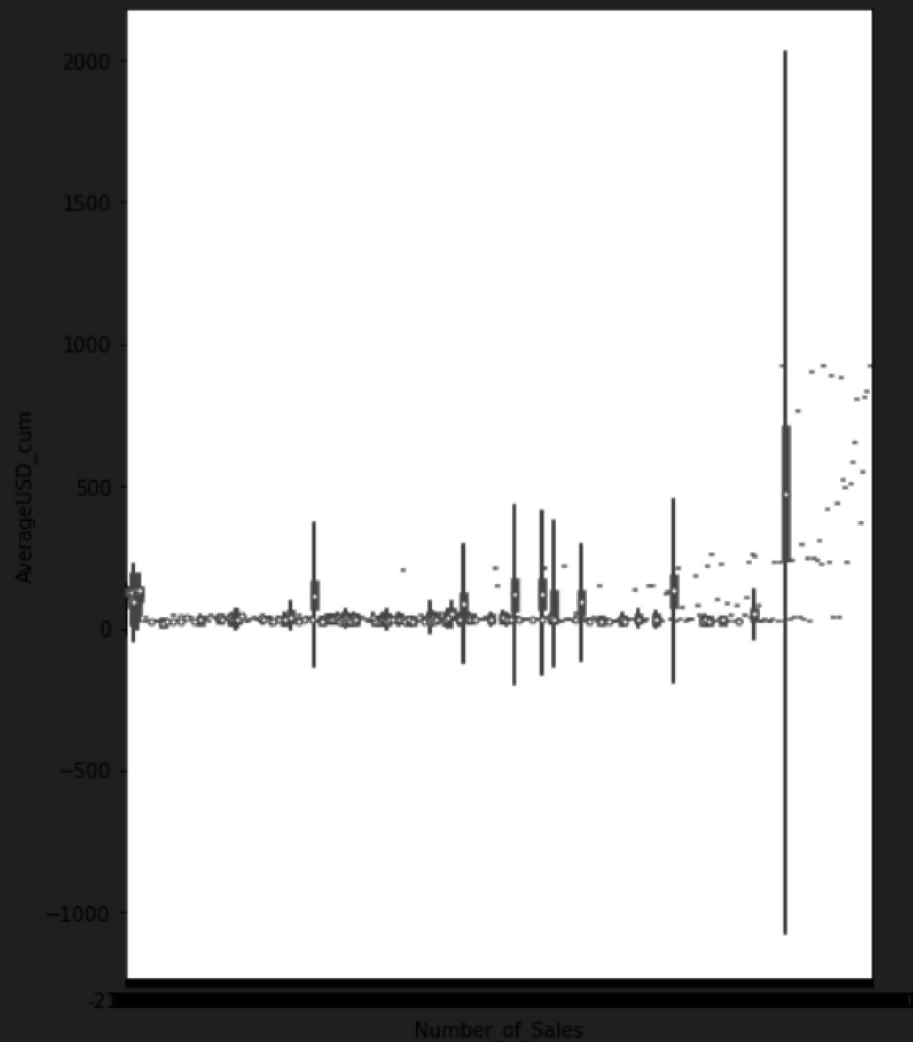



```
df.hist('AverageUSD_cum')
```

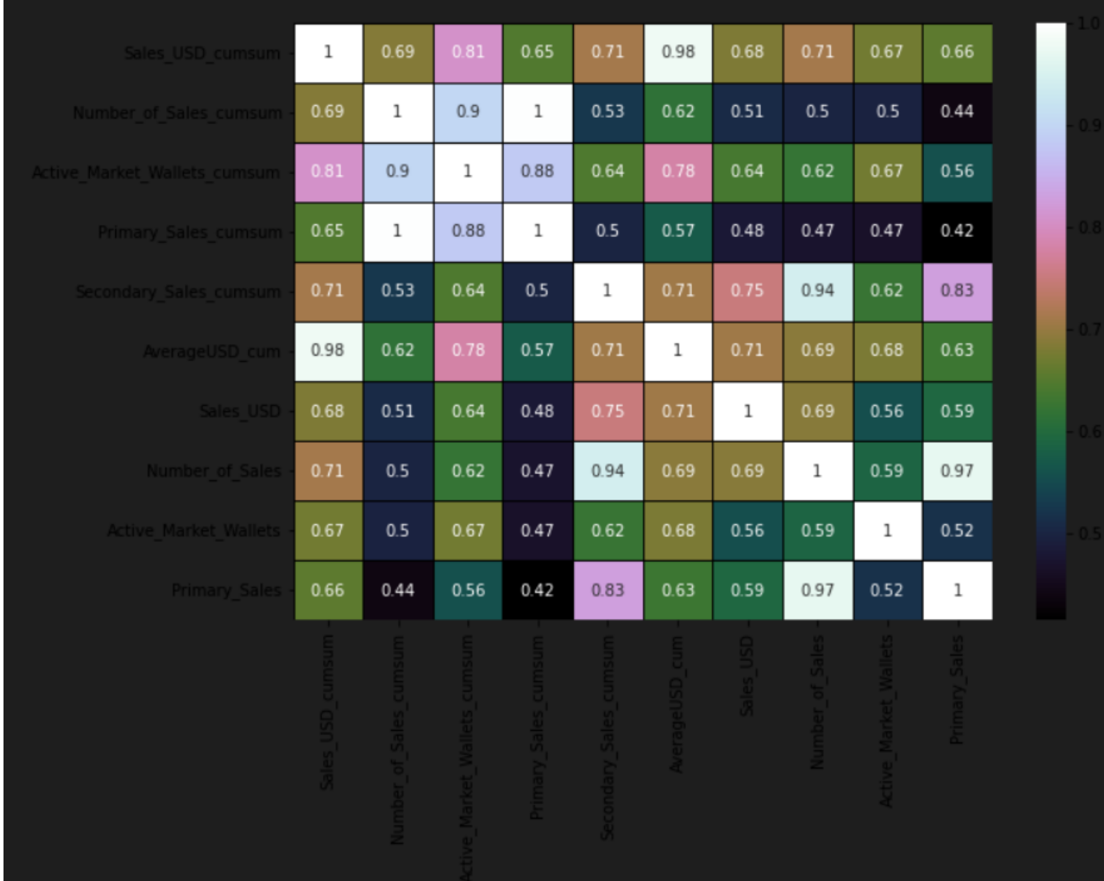
```
array([[<AxesSubplot:title={'center':'AverageUSD_cum'}>]], dtype=object)
```



```
plt.figure(figsize=(15,20))
plt.subplot(2,2,1)
sns.violinplot(x='Number_of_Sales',y='AverageUSD_cum',data=df)
plt.show()
```



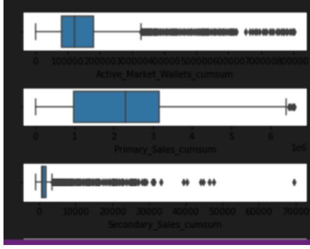
```
fig=plt.gcf()
fig.set_size_inches(10,7)
fig=sns.heatmap(df.corr(),annot=True,cmap='cubehelix',linewidths=1,linecolor='k')
```



```python
import seaborn as sns
fig, axs = plt.subplots(4, figsize = (5,5))
plt1 = sns.boxplot(df['Active_Market_Wallets_cumsum'], ax = axs[0])
plt2 = sns.boxplot(df['Primary_Sales_cumsum'], ax = axs[1])
plt3 = sns.boxplot(df['Secondary_Sales_cumsum'], ax = axs[2])
plt4 = sns.boxplot(df['AverageUSD_cum'], ax = axs[3])
plt.tight_layout()
```

Python

```
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```




```python
#from sklearn.linear_model import LinearRegression
#model = LinearRegression()
#model.fit(x, y)
#y_pred = model.predict(x_test)
#confidence = model.score(x_test, y_test)
#print(confidence)
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.5,random_state=42)
```

```python
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor = regressor.fit(x_train, y_train)
y_pred = regressor.predict(x_test)
print(y_pred)
confidence = regressor.score(x_train,y_pred)
print("Acuuracy of the model:  ",confidence)
```

```
Output exceeds the size limit. Open the full output data in a text editor
[ 9.75059413e+00  1.00011849e+02  2.23417494e+02  4.73576150e+01
  5.58479406e+01  4.49104126e+02  1.67388753e+02  8.95221066e+01
  2.28021628e+01  7.47386245e+01  1.77903831e+01  6.01155225e+01
 -3.89537762e+01  7.08048304e+02  5.10592237e+00  4.72816676e+01
  4.72500893e+01  1.42478315e+02  5.74980980e+01  6.50844993e+01
  1.18597122e+02 -4.13875939e+00  5.02756467e+02  9.94889227e+01
  4.72745354e+01  1.57446975e+01  3.06729208e+00 -5.00492606e+01
  6.96808290e+01  8.25706831e+02  4.72870879e+01  5.41160037e+01
  1.20190712e+02  1.70164360e+02  1.11819892e+02  3.63004754e+00
  1.58026471e+01  8.78187483e+01  1.12488483e+02  1.97971170e+02
  8.79490448e+01  2.01081920e+01 -5.84222349e-01  4.71937264e+01
  5.95732497e+01  4.72791282e+01  5.14813020e+01  4.73162024e+01
 -4.00089260e+01  1.08928416e+02  3.63703211e+01  1.05639185e+02
  6.86988254e+02  5.47968998e+01  6.84918518e+01  2.01098924e+02
  9.16249776e+01  6.31574120e+01 -4.08807963e+00 -4.94909505e+01
  1.16069905e+02  4.73063094e+01  1.05813917e+02  1.07589138e+02
 -3.72618570e+01  4.72756298e+01  1.70020018e+02  7.55232580e+01
 -3.45380969e+01 -1.04802966e+01  1.17597279e+02 -1.90695145e+01
  1.15206359e+02  1.22880234e+02 -2.99634064e+01  1.14531921e+02
  1.07233616e+02  5.62590076e+01  3.74137572e+01  4.91087064e+02
 -3.38965492e+01  7.76572303e+01  1.11270043e+02  1.14942804e+02
 -3.84761418e+01  1.04823912e+02  3.83601971e+01  6.52122766e+02
  1.42619462e+02 -6.32201696e+00  1.81003076e+02  9.16627399e+02
  5.38816549e+01  1.21442862e+02  6.03063670e+01 -3.60762113e+01
  7.65996660e+01  4.72586594e+01  1.01229726e+02 -5.33161686e+01
...
  1.13868091e+02  4.73062417e+01  2.24467634e+02  4.73018678e+01
 -2.54964988e+01 -4.23072888e+01  9.12919074e+01  7.76860330e+01
 -3.34592580e+01  5.90164937e+01  1.15025225e+01]
Acuuracy of the model:   -1.0260191418952598
```

```python
X = df.drop(['AverageUSD_cum','Date'],axis=1).values
X[0:5]
```

```
array([[   0.        ,    0.        ,    0.        ,    0.        ,
           0.        ,    0.        ,    0.        ,  502.87539136,
           0.        ],
       [1020.3        ,   19.        ,    8.        ,    0.        ,
          19.        , 1020.3        ,   19.        ,    8.        ,
           0.        ],
       [2261.14       ,   40.        ,   21.        ,    0.        ,
          21.        , 1240.84       ,   21.        ,   13.        ,
           0.        ],
       [2778.69       ,   53.        ,   28.        ,    0.        ,
          13.        ,  517.55       ,   13.        ,    7.        ,
           0.        ],
       [3203.32       ,   67.        ,   34.        ,    0.        ,
          14.        ,  424.63       ,   14.        ,    6.        ,
           0.        ]])
```

```python
Y = df['AverageUSD_cum'].values
```

```python
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=1)
```

```python
regressor = LinearRegression()
regressor.fit(X_train, Y_train)
```

```
LinearRegression()
```

```python
Y_predict = regressor.predict(X_test)
Y_predict[0:20]
```

```
array([ 39.29516599,  20.18992732,  31.68471214,  48.0839771 ,
        25.43790456, 646.34956689, 118.75525   ,  42.56130434,
        69.46549542,  35.33582055, 121.57233165,  40.50819454,
        29.92785325,  32.59788236, 682.73126896,  34.1982961 ,
       911.37671386,  67.99818673,  31.15291224,  60.7089928 ])
```

```python
from sklearn.metrics import r2_score
r2_score(Y_predict, Y_test)
```

```
0.9675095771486381
```

```python
from sklearn.metrics import r2_score
r2_score(Y_predict, Y_test)
```

```
0.9675095771486381
```

```python
score = regressor.score(x_test, y_test)
score
```

```
0.9701773034691825
```