



NFTLYZE - REAL-TIME ANALYSIS OF NFT MARKET TRENDS – PROJECT REPORT

BY GOPIKA LAKSHMY

[GitHub Repository Link.](#)

Introduction

The NFT (Non-Fungible Token) market has grown rapidly, leading to an increased demand for real-time data analysis tools to track trends in the NFT space. This project aims to develop a Python-based dashboard using Dash and Plotly to visualize key metrics in the NFT market, such as sales volume, average price, and active wallets over time.

This dashboard can be used to monitor market trends, analyze NFT performance, and assist in decision-making for both collectors and investors.

Objectives

The main objectives of this project are:

- To build an interactive dashboard that visualizes NFT market trends using real-time data.
- To provide users with insights into sales volume, average price, and wallet activity.
- To create a user-friendly interface that allows for data filtering and exploration.
- To enable future integration with live NFT marketplace data, such as OpenSea API.

Methodology

Data Collection

The initial data for this project was simulated using Python to represent various NFT metrics over time. This included:

- Date: The time series of data collection.
- Sales_USD: The total sales volume in USD for the given time period.
- Average_Price: The average price of NFTs sold during that time.
- Volume_USD: The total transaction volume in USD.
- Active_Wallets: The number of unique wallets active in the marketplace.

The dashboard is built to allow for easy integration with real-time NFT data sources, including APIs from major marketplaces like OpenSea.

Dashboard Development

The dashboard was developed using Dash, a Python framework for building web applications, and Plotly, a data visualization library. The dashboard displays various NFT market trends through line charts and scatter plots. The key features include:

- Dropdown selection: Users can select between different metrics (e.g., Sales, Average Price, Volume, Active Wallets) to view over time.

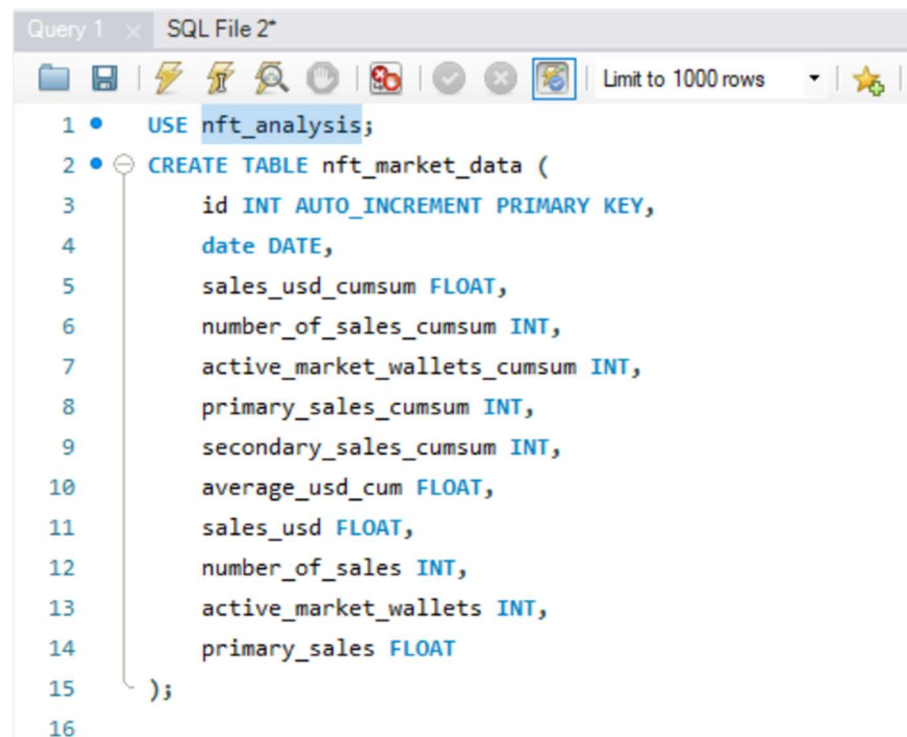
- Interactive graphs: The graphs update dynamically based on user inputs.

Tools and Technologies

- Dash: A Python framework for web-based interactive dashboards.
- Plotly: For creating interactive plots and charts.
- Pandas: For data manipulation and handling.
- GitHub: Used for version control and project hosting.
- MySQL: The real-time data integration feature was designed to store data in a MySQL database (optional).

Methodology:

1. Create a MySQL database (e.g., nft_analysis).



The screenshot shows a SQL IDE window with a tab labeled 'SQL File 2*'. The query editor contains the following SQL code:

```
1 • USE nft_analysis;
2 • CREATE TABLE nft_market_data (
3     id INT AUTO_INCREMENT PRIMARY KEY,
4     date DATE,
5     sales_usd_cumsum FLOAT,
6     number_of_sales_cumsum INT,
7     active_market_wallets_cumsum INT,
8     primary_sales_cumsum INT,
9     secondary_sales_cumsum INT,
10    average_usd_cum FLOAT,
11    sales_usd FLOAT,
12    number_of_sales INT,
13    active_market_wallets INT,
14    primary_sales FLOAT
15 );
16
```

```

Query 1 x SQL File 2*
Limit to 1000 rows

17 CREATE TABLE nft_additional_data (
18     id INT AUTO_INCREMENT PRIMARY KEY,
19     volume_usd FLOAT,
20     market_cap FLOAT,
21     market_cap_usd FLOAT,
22     sales INT,
23     floor_price FLOAT,
24     floor_price_usd FLOAT,
25     average_price FLOAT,
26     average_price_usd FLOAT,
27     owners INT,
28     assets INT,
29     owner_asset_ratio FLOAT,
30     category VARCHAR(255),
31     website VARCHAR(255),
32     logo VARCHAR(255)
33 );

```

2. Prepare a CSV data in Python using pandas.

```

[13]: import pandas as pd

# Load the datasets
first_dataset = pd.read_csv(r"C:\Users\gopik\OneDrive\Desktop\upgrad\projects\NFT Analysis\NFTlyze Dataset.csv")
second_dataset = pd.read_csv(r"C:\Users\gopik\OneDrive\Desktop\upgrad\projects\NFT Analysis\top 10 nft.csv")

# Display the data to check
print(first_dataset.head())
print(second_dataset.head())

```

	Date	Sales_USD_cumsum	Number_of_Sales_cumsum	\
0	22-06-2017	0.00	0.0	
1	23-06-2017	1020.30	19.0	
2	24-06-2017	2261.14	40.0	
3	25-06-2017	2778.69	53.0	
4	26-06-2017	3203.32	67.0	

	Active_Market_Wallets_cumsum	Primary_Sales_cumsum	Secondary_Sales_cumsum	\
0	0.0	0.0	0.0	
1	8.0	0.0	19.0	
2	21.0	0.0	21.0	
3	28.0	0.0	13.0	
4	34.0	0.0	14.0	

3. Connect Python to MySQL using SQLAlchemy and pymysql.

```
[17]: from sqlalchemy import create_engine

# Connection string format
engine = create_engine('mysql+pymysql://root:gopika%40mysql@localhost/nft_analysis')

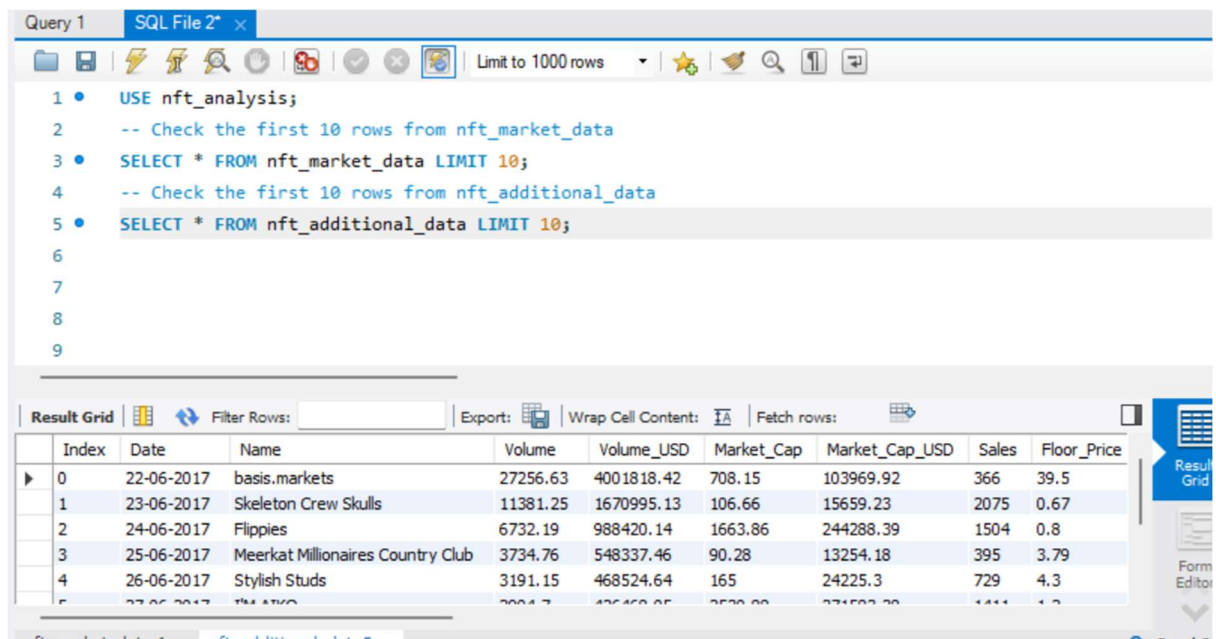
[19]: first_dataset.to_sql('nft_market_data', con=engine, if_exists='replace', index=False)

[19]: 1606

[21]: second_dataset.to_sql('nft_additional_data', con=engine, if_exists='replace', index=False)

[21]: 592
```

4. Upload your data into MySQL using the `to_sql()` function. Verify the data and perform further analysis.



The screenshot shows a SQL client window titled "Query 1" and "SQL File 2". The query editor contains the following SQL code:

```
1 • USE nft_analysis;
2 -- Check the first 10 rows from nft_market_data
3 • SELECT * FROM nft_market_data LIMIT 10;
4 -- Check the first 10 rows from nft_additional_data
5 • SELECT * FROM nft_additional_data LIMIT 10;
```

The results are displayed in a table with the following columns: Index, Date, Name, Volume, Volume_USD, Market_Cap, Market_Cap_USD, Sales, and Floor_Price. The first 10 rows of data are shown.

Index	Date	Name	Volume	Volume_USD	Market_Cap	Market_Cap_USD	Sales	Floor_Price
0	22-06-2017	basis.markets	27256.63	4001818.42	708.15	103969.92	366	39.5
1	23-06-2017	Skeleton Crew Skulls	11381.25	1670995.13	106.66	15659.23	2075	0.67
2	24-06-2017	Flippies	6732.19	988420.14	1663.86	244288.39	1504	0.8
3	25-06-2017	Meerkat Millionaires Country Club	3734.76	548337.46	90.28	13254.18	395	3.79
4	26-06-2017	Stylish Studs	3191.15	468524.64	165	24225.3	729	4.3
5	27-06-2017	THATCO	2001.7	436460.05	2520.00	371522.30	1411	4.3

```
[27]: # Query data from the nft_market_data table
query = "SELECT * FROM nft_market_data"
df = pd.read_sql(query, con=engine)

# Display the DataFrame
print(df.head())
```

	Date	Sales_USD_cumsum	Number_of_Sales_cumsum	\
0	22-06-2017	0.00	0.0	
1	23-06-2017	1020.30	19.0	
2	24-06-2017	2261.14	40.0	
3	25-06-2017	2778.69	53.0	
4	26-06-2017	3203.32	67.0	

	Active_Market_Wallets_cumsum	Primary_Sales_cumsum	Secondary_Sales_cumsum	\
0	0.0	0.0	0.0	
1	8.0	0.0	19.0	
2	21.0	0.0	21.0	
3	28.0	0.0	13.0	
4	34.0	0.0	14.0	

Exploratory Data Analysis (EDA)

To perform Exploratory Data Analysis (EDA) on your NFT sales data, we will focus on analyzing the distribution of NFT sales across different marketplaces and over time. This will help us understand the patterns, trends, and outliers in the data.

Load the Data

We've already uploaded your data into MySQL and can now use Python to retrieve it for analysis.

```
[29]: import pandas as pd

# Query data from MySQL tables
nft_market_data = pd.read_sql("SELECT * FROM nft_market_data", con=engine)
nft_additional_data = pd.read_sql("SELECT * FROM nft_additional_data", con=engine)

# Take a Look at the data
print(nft_market_data.head())
print(nft_additional_data.head())
```

	Date	Sales_USD_cumsum	Number_of_Sales_cumsum	\
0	22-06-2017	0.00	0.0	
1	23-06-2017	1020.30	19.0	
2	24-06-2017	2261.14	40.0	
3	25-06-2017	2778.69	53.0	
4	26-06-2017	3203.32	67.0	

	Active_Market_Wallets_cumsum	Primary_Sales_cumsum	Secondary_Sales_cumsum	\
0	0.0	0.0	0.0	
1	8.0	0.0	19.0	
2	21.0	0.0	21.0	
3	28.0	0.0	13.0	
4	34.0	0.0	14.0	

	AverageUSD_cum	Sales_USD	Number_of_Sales	Active_Market_Wallets	\
0	0.00	0.00	0.0	0.0	
1	53.70	1020.30	19.0	8.0	

Understand the Structure of the Data

Let's explore the data to understand its structure, such as the types of columns, missing values, and general statistics. This will give us a quick overview of the dataset, including data types, missing values, and summary statistics like mean, median, and standard deviation.

```
[31]: # Basic info on the dataset
nft_market_data.info()
nft_additional_data.info()

# Check for missing values
print(nft_market_data.isnull().sum())
print(nft_additional_data.isnull().sum())

# Summary statistics
print(nft_market_data.describe())
print(nft_additional_data.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1606 entries, 0 to 1605
Data columns (total 11 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Date                                592 non-null    object
1   Sales_USD_cumsum                    592 non-null    float64
2   Number_of_Sales_cumsum              592 non-null    float64
3   Active_Market_Wallets_cumsum        592 non-null    float64
4   Primary_Sales_cumsum                592 non-null    float64
5   Secondary_Sales_cumsum               592 non-null    float64
6   AverageUSD_cum                     592 non-null    float64
7   Sales_USD                          592 non-null    float64
8   Number of Sales                    592 non-null    float64
```

Analyze NFT Sales Over Time

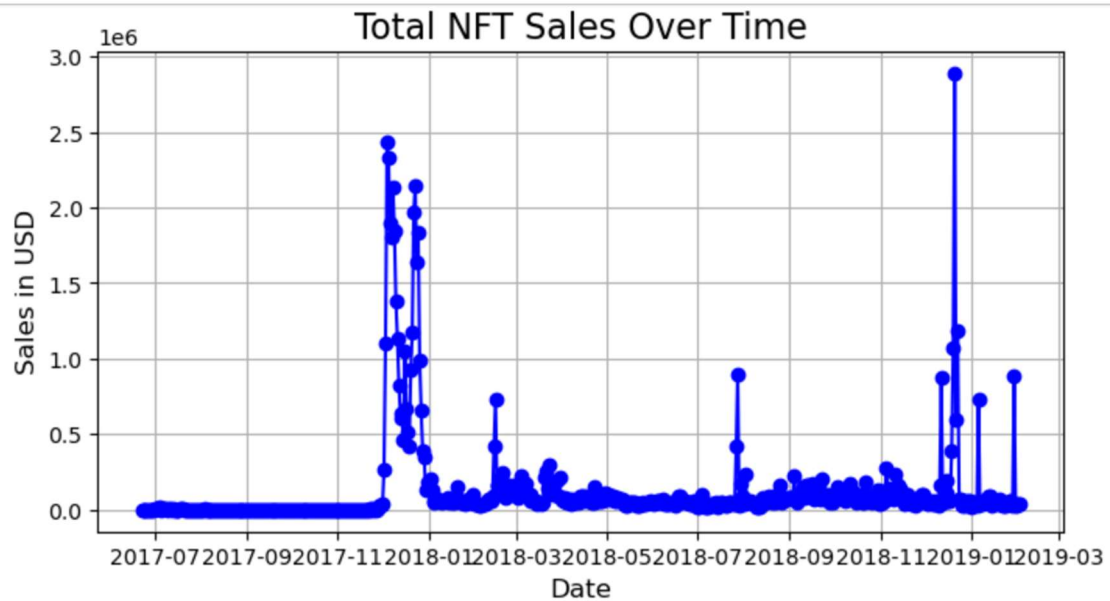
To analyze sales over time, you can group the data by Date and visualize the total sales.

```
[33]: import matplotlib.pyplot as plt

# Convert 'Date' to datetime format
nft_market_data['Date'] = pd.to_datetime(nft_market_data['Date'])

# Group the data by date and sum the sales
sales_over_time = nft_market_data.groupby('Date')['Sales_USD'].sum().reset_index()

# Plot total sales over time
plt.figure(figsize=(10, 6))
plt.plot(sales_over_time['Date'], sales_over_time['Sales_USD'], color='blue', marker='o')
plt.title('Total NFT Sales Over Time', fontsize=16)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Sales in USD', fontsize=12)
plt.grid(True)
plt.show()
```

Key Observations:

1. Time Period: The x-axis represents the date range, starting from mid-2017 to early 2019.
 - The dates are spaced over several months, suggesting that the data covers a continuous period.
2. Sales in USD:
 - The y-axis represents the total sales in USD, with values reaching up to 2.5 million.
 - The sales values are represented in scientific notation (e.g., $1e6 = 1,000,000$ USD).
3. Initial Period (Flat Line):
 - From mid-2017 to late 2017, there's minimal activity or almost zero sales in USD. This could suggest that the NFT market was either just starting or there was low market activity during this period.
4. Spike in Sales (Late 2017 to Early 2018):
 - A sharp spike occurs towards the end of 2017 and continues into early 2018, where total sales reach up to 2.5 million USD.
 - This sudden spike suggests a significant increase in NFT market activity or a potential event that led to this surge (e.g., a popular NFT release, a broader market trend, or hype).
5. Post-Spike Decline:

- After the initial spike, sales decline rapidly, returning to lower levels by early 2018. This could indicate that the market experienced a bubble or a correction after a period of intense activity.

6. Smaller Spikes:

- Throughout the rest of 2018 and early 2019, smaller spikes in sales can be observed, although they do not reach the heights of the earlier period.
- This suggests that while the NFT market remains active, no event or collection has matched the earlier spike in popularity or financial volume.

7. Overall Trend:

- The overall trend after the large spike is much flatter, suggesting a steady but lower level of market activity for the rest of the observed period.

Possible Interpretations:

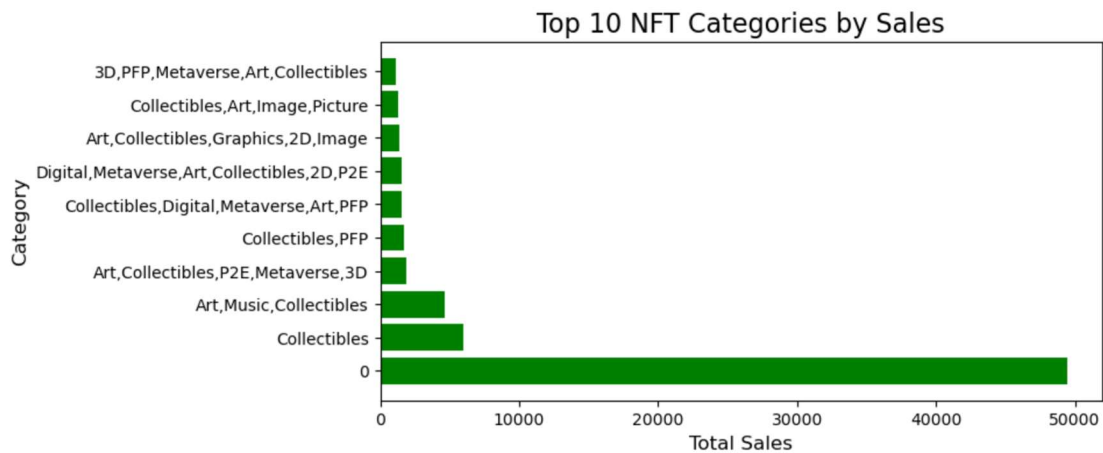
- Initial Market Growth: The initial flat line indicates a relatively quiet period for NFTs. This could align with the early days of NFT technology before it gained mainstream attention.
- Market Hype or Event: The massive spike suggests a boom period, likely driven by high-profile releases or a surge in NFT popularity in late 2017 to early 2018.
- Market Correction: The sharp drop after the peak could signify a cooling off or market correction after the initial hype phase.
- Sustained Activity: Despite the fall after the initial peak, smaller spikes show that the market remained active but not as volatile, indicating a stabilization of sorts.

Analyze Sales Distribution Across Marketplaces

- Assuming that the `nft_additional_data` contains information about different marketplaces (categories), we can explore how the sales are distributed across top 10 categories.

```
# Sort by total sales and select the top 10 categories
top_sales_by_category = sales_by_category.sort_values(by='Sales', ascending=False).head(10)

# Plot the top 10 categories
plt.figure(figsize=(8, 4))
plt.barh(top_sales_by_category['Category'], top_sales_by_category['Sales'], color='green')
plt.title('Top 10 NFT Categories by Sales', fontsize=16)
plt.xlabel('Total Sales', fontsize=12)
plt.ylabel('Category', fontsize=12)
plt.show()
```



Key Observations:

1. **Dominant Category:** The category labeled "Collectibles" far surpasses all other categories in total sales, with nearly 50,000 sales. It is clearly the top performer in the NFT space by a significant margin.
2. **Smaller Categories:** Other categories, such as "Art, Music, Collectibles" and "Art, Collectibles, P2E, Metaverse, 3D", have much lower sales, with totals around 10,000 and less.
3. **Many Overlapping Categories:** The categories seem to combine various NFT types (e.g., "Art", "Collectibles", "Metaverse", "PFP", etc.), but many of them overlap in terms of their components, indicating that these segments are often cross-categorized.

In summary, the chart reveals a massive concentration of sales in the "Collectibles" category, while the other categories are significantly less impactful in terms of total sales.

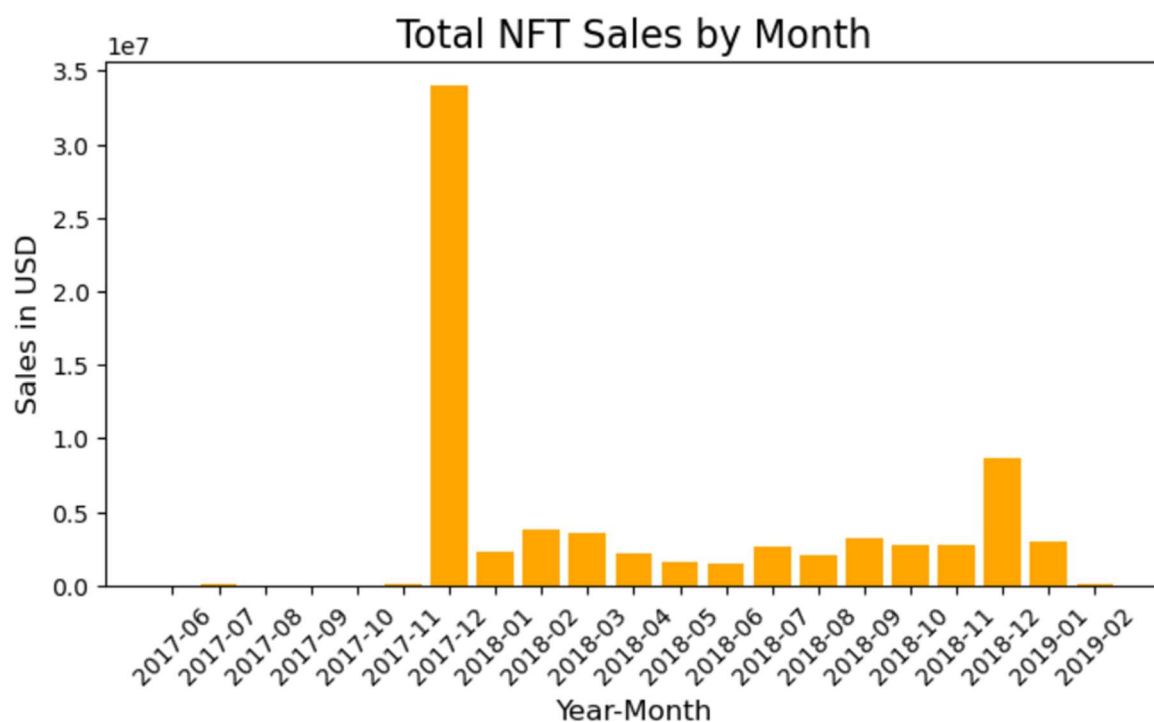
Analyze Sales Distribution by Time Periods

We can also break down the sales data into various time periods, such as months, quarters, or years, to identify seasonal trends or specific periods with higher sales activity.

```
# Extract year and month from the 'Date' column
nft_market_data['YearMonth'] = nft_market_data['Date'].dt.to_period('M')

# Group by year and month and calculate total sales
sales_by_period = nft_market_data.groupby('YearMonth')['Sales_USD'].sum().reset_index()

# Plot total sales by month
plt.figure(figsize=(8, 4))
plt.bar(sales_by_period['YearMonth'].astype(str), sales_by_period['Sales_USD'], color='orange')
plt.title('Total NFT Sales by Month', fontsize=16)
plt.xlabel('Year-Month', fontsize=12)
plt.ylabel('Sales in USD', fontsize=12)
plt.xticks(rotation=45)
plt.show()
```



Key Observations:

1. Major Sales Spike in November 2017:
 - There is a significant spike in NFT sales in November 2017, with sales reaching around 3.5 million USD. This is by far the highest point on the chart, indicating a major event or surge in the NFT market during this period.
2. Moderate Activity Following the Spike:
 - After the November 2017 peak, sales decrease sharply in the following months but remain somewhat active throughout early 2018, with smaller sales fluctuations.

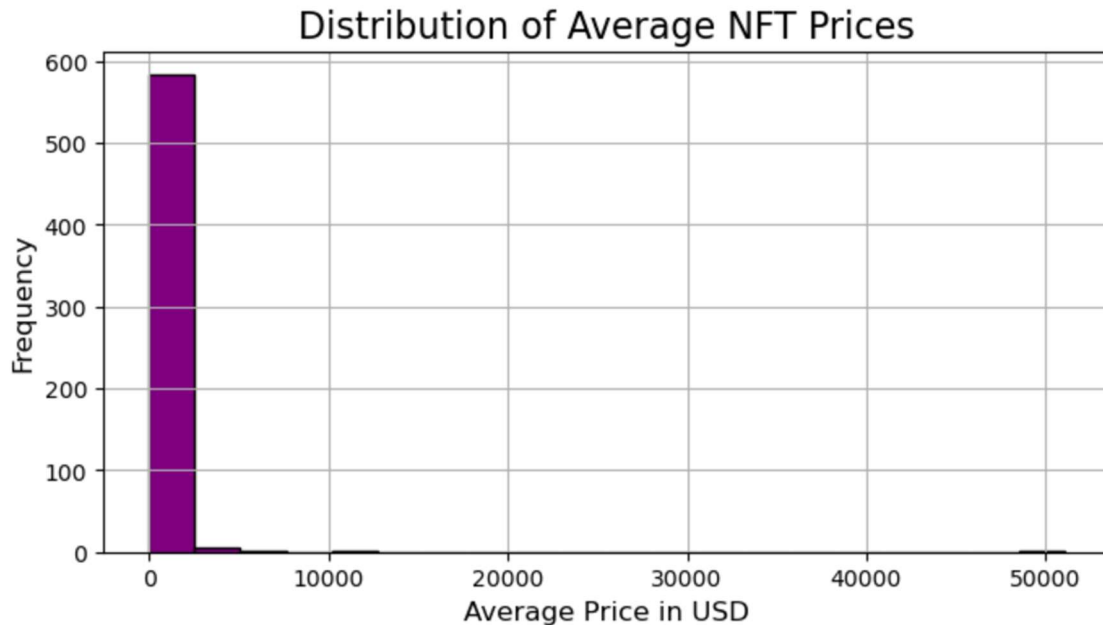
- Another minor peak occurs in November 2018, but it's significantly smaller than the November 2017 surge.
3. Low Sales Before and After Peaks:
- Before the surge in November 2017 and after early 2019, sales remain very low, indicating periods of minimal market activity or growth.
4. Sustained Lower Sales in 2018:
- For most of 2018, sales continue at a much lower level compared to the November 2017 spike. This may suggest a market correction after the initial boom.

In summary the chart highlights a massive sales event in November 2017, followed by a steady decline in activity with some fluctuations throughout 2018. The market shows occasional upticks but never reaches the same level as the peak in late 2017. This could indicate an initial burst of interest in NFTs followed by market stabilization.

Analyze Distribution of Prices

You can analyze the distribution of NFT prices to understand how the prices vary across marketplaces or time periods.

```
# Plot a histogram of average prices
plt.figure(figsize=(8, 4))
plt.hist(nft_additional_data['Average_Price_USD'], bins=20, color='purple', edgecolor='black')
plt.title('Distribution of Average NFT Prices', fontsize=16)
plt.xlabel('Average Price in USD', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.grid(True)
plt.show()
```



The histogram titled "Distribution of Average NFT Prices" shows that the vast majority of NFTs have an average price concentrated around 0 to 10,000 USD, with a particularly high frequency below 1,000 USD. Only a very small number of NFTs are priced above 10,000 USD, and the distribution tails off significantly after that, indicating that most NFTs are sold at relatively low prices. The chart highlights a sharp skew towards lower-priced NFTs.

Correlation Analysis

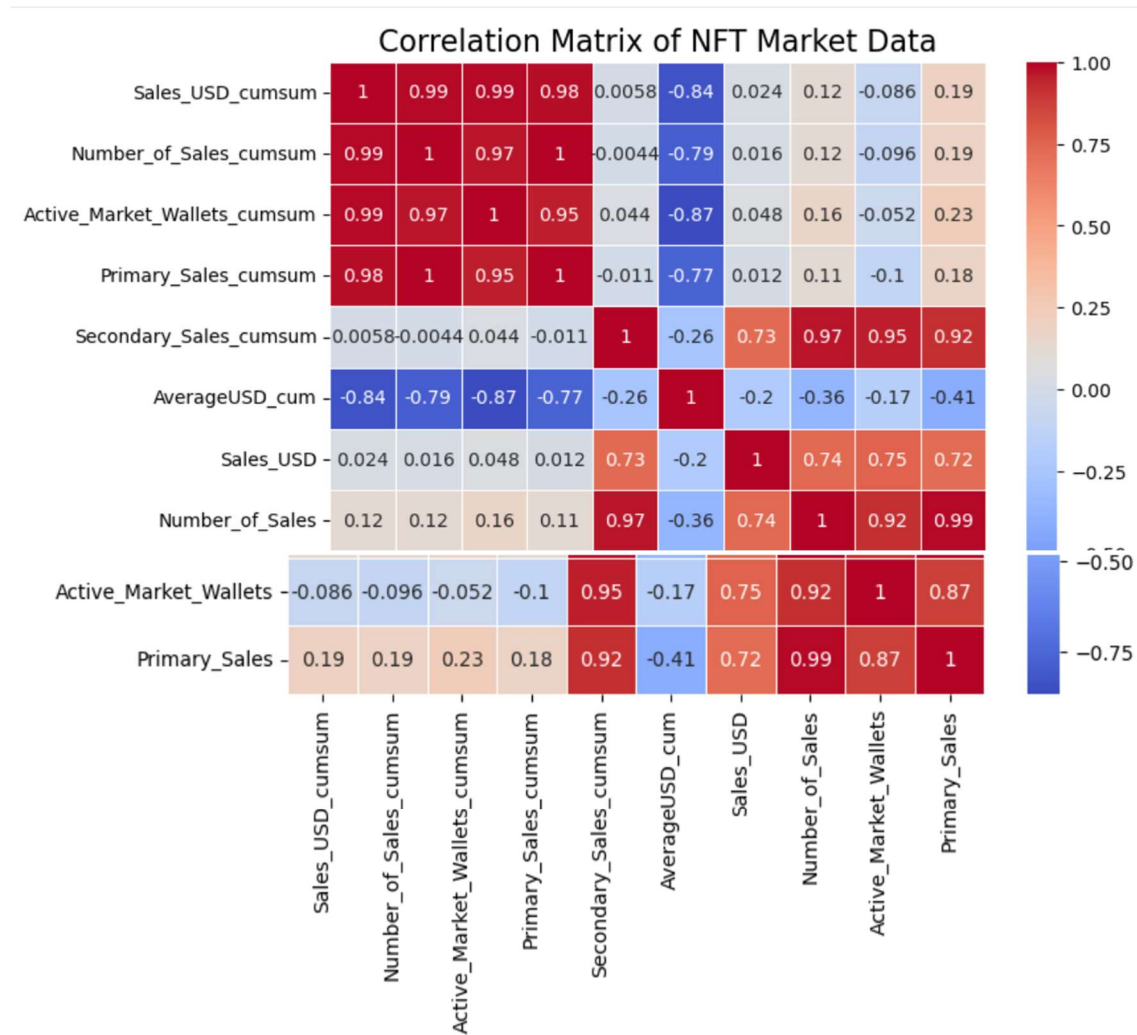
we can perform a correlation analysis between various numeric features in the dataset to see if there are any strong relationships.

```
: # Select only numeric columns for correlation analysis
numeric_columns = nft_market_data.select_dtypes(include=['float64', 'int64'])

# Calculate correlation matrix
correlation_matrix = numeric_columns.corr()

# Plot the heatmap of the correlation matrix
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix of NFT Market Data', fontsize=16)
plt.show()
```



Key Observations:

1. Highly Correlated Pairs:

- Sales_USD_cumsum and Number_of_Sales_cumsum (0.99): This near-perfect positive correlation indicates that the cumulative sales amount in USD is almost directly proportional to the number of sales. More sales typically result in a higher cumulative sales value.
- Primary_Sales_cumsum and Sales_USD_cumsum (0.98): The primary sales cumulative value is also highly correlated with cumulative sales in USD.
- Sales_USD_cumsum and Active_Market_Wallets_cumsum (0.99): This strong positive correlation indicates that as the number of active market wallets increases, the total USD sales also increase.

2. Low or Negative Correlations:

- Sales_USD_cumsum and AverageUSD_cum (-0.77): This negative correlation suggests that as cumulative sales in USD increase, the average sale value decreases, implying that there may be a large volume of low-priced sales driving the total sales.
- Secondary_Sales_cumsum and Sales_USD_cumsum (-0.006): This very weak correlation shows that secondary sales do not have a strong relationship with the cumulative sales in USD.

3. Strong Positive Correlations:

- The upper diagonal part of the matrix (shaded in dark red) indicates strong positive correlations among the cumulative columns, including Number_of_Sales_cumsum, Active_Market_Wallets_cumsum, and Primary_Sales_cumsum.

4. AverageUSD_cum:

- The variable AverageUSD_cum is negatively correlated with most cumulative metrics, suggesting that as the average price per sale decreases, the overall market activity (in terms of the number of sales and total wallets) increases.

Analyze Top 10 NFTs by Price:

```
# Sort by Average_Price_USD to find top-priced NFTs
top_nfts_by_price = nft_data_combined.sort_values(by='Average_Price_USD', ascending=False).head(10)

# Display relevant columns for analysis
print(top_nfts_by_price[['Name', 'Category', 'Average_Price_USD', 'Sales_USD']])
```


	Name \		
25	SolanaMonkeyBusiness		
0	basis.markets		
99	The Exiled Apes		
92	Aurory		
46	Electric Blue		
114	Turtles		
81	Shapes of Reality		
286	Numoon		
213	Degenerate Ape Academy		
16	Psychedelic Apes		
		Category	Average_Price_USD \
25		0	51019.95
0		Collectibles,Digital,Privilege	10933.93
99		0	7243.12
92		0	6221.13
46	Art,Collectibles,Photography,Metaverse,Perform...		5018.57
114		0	4478.01
81	Digital,Art,Collectibles,Immaterial,Metaverse,...		3780.62
286	Digital,2D,Art,Collectibles		2936.40
213		0	2866.66
16	Digital,Art,Metaverse,Collectibles,PFP,3D		2401.31

Trend Analysis and Forecasting:

Prepare the Data

The first step is to prepare the dataset for time series forecasting. We need to ensure that the data is properly indexed by Date and cleaned for any missing values.

```
# Convert 'YearMonth' from Period to datetime (timestamps)
nft_market_data['Date'] = nft_market_data['YearMonth'].dt.to_timestamp()

# Set 'Date' as the index
nft_market_data.set_index('Date', inplace=True)

# Drop 'YearMonth' as it's no longer needed
nft_market_data.drop(columns=['YearMonth'], inplace=True)

# Check if the conversion is successful
print(nft_market_data.head())
```

	Sales_USD_cumsum	Number_of_Sales_cumsum	\
Date			
2017-06-01	0.00	0.0	
2017-06-01	1020.30	19.0	
2017-06-01	2261.14	40.0	
2017-06-01	2778.69	53.0	
2017-06-01	3203.32	67.0	

	Active_Market_Wallets_cumsum	Primary_Sales_cumsum	\
Date			
2017-06-01	0.0	0.0	
2017-06-01	8.0	0.0	
2017-06-01	21.0	0.0	
2017-06-01	28.0	0.0	
2017-06-01	34.0	0.0	

	Secondary_Sales_cumsum	AverageUSD_cum	Sales_USD	\
Date				
2017-06-01	0.0	0.00	0.00	
2017-06-01	19.0	53.70	1020.30	
2017-06-01	21.0	56.53	1240.84	
2017-06-01	13.0	52.43	517.55	
2017-06-01	14.0	47.81	424.63	

ARIMA Model

Fit an ARIMA model:

ARIMA (AutoRegressive Integrated Moving Average) is a popular model for time series forecasting. We will use it to forecast NFT sales.

```

import statsmodels.api as sm
from statsmodels.tsa.arima.model import ARIMA

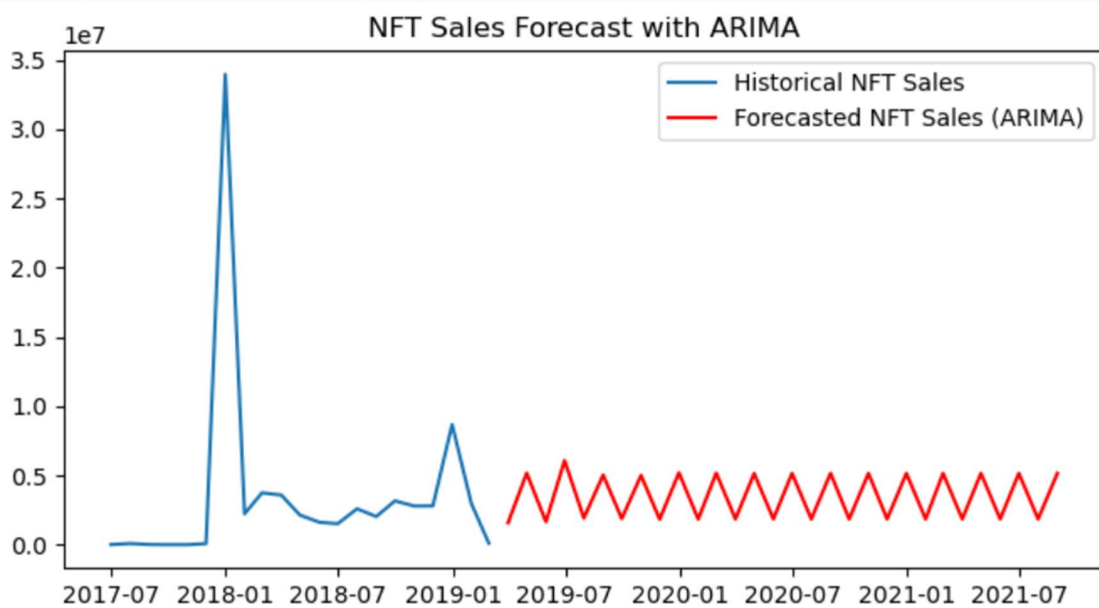
# Use the 'monthly_sales' data from previous steps
# Fit an ARIMA model (p, d, q values can be tuned)
model_arima = ARIMA(monthly_sales, order=(5, 1, 2)) # Example order (p, d, q)
arima_result = model_arima.fit()

# Forecast the next 30 periods (months)
forecast_arima = arima_result.forecast(steps=30)

# Plot the original sales data and forecast
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
plt.plot(monthly_sales.index, monthly_sales, label="Historical NFT Sales")
plt.plot(forecast_arima.index, forecast_arima, label="Forecasted NFT Sales (ARIMA)", color='red')
plt.legend()
plt.title("NFT Sales Forecast with ARIMA")
plt.show()

# Print summary of ARIMA model
print(arima_result.summary())

```



```

=====
SARIMAX Results
=====
Dep. Variable:          Sales_USD      No. Observations:          21
Model:                 ARIMA(5, 1, 2)  Log Likelihood             -364.998
Date:                  Fri, 04 Oct 2024 AIC                          745.996
Time:                  15:43:23       BIC                         753.962
Sample:                06-30-2017     HQIC                        747.551
                  - 02-28-2019
Covariance Type:       opg
=====
              coef    std err          z      P>|z|      [0.025    0.975]
-----
ar.L1         -1.0052      0.123     -8.159      0.000     -1.247     -0.764
ar.L2         -0.1346      0.218     -0.617      0.537     -0.563      0.293
ar.L3         -0.1241      0.277     -0.448      0.654     -0.667      0.419
ar.L4         -0.1779      0.492     -0.361      0.718     -1.143      0.787
ar.L5         -0.1832      0.389     -0.471      0.638     -0.946      0.579
ma.L1          -0.0009      0.005     -0.169      0.866     -0.012      0.010
ma.L2         -0.9991      0.110     -9.122      0.000     -1.214     -0.784
sigma2         1.098e+13   9.72e-15   1.13e+27      0.000     1.1e+13     1.1e+13
=====
Ljung-Box (L1) (Q):                0.04   Jarque-Bera (JB):                125.15
Prob(Q):                            0.85   Prob(JB):                      0.00

```

Prophet Model

Prophet is another widely used model for time series forecasting developed by Facebook. It works well for data with daily seasonality, holidays, and trends.

```

: from prophet import Prophet

# Prepare the monthly sales data for Prophet
prophet_data = monthly_sales.reset_index().rename(columns={'Date': 'ds', 'Sales_USD': 'y'})

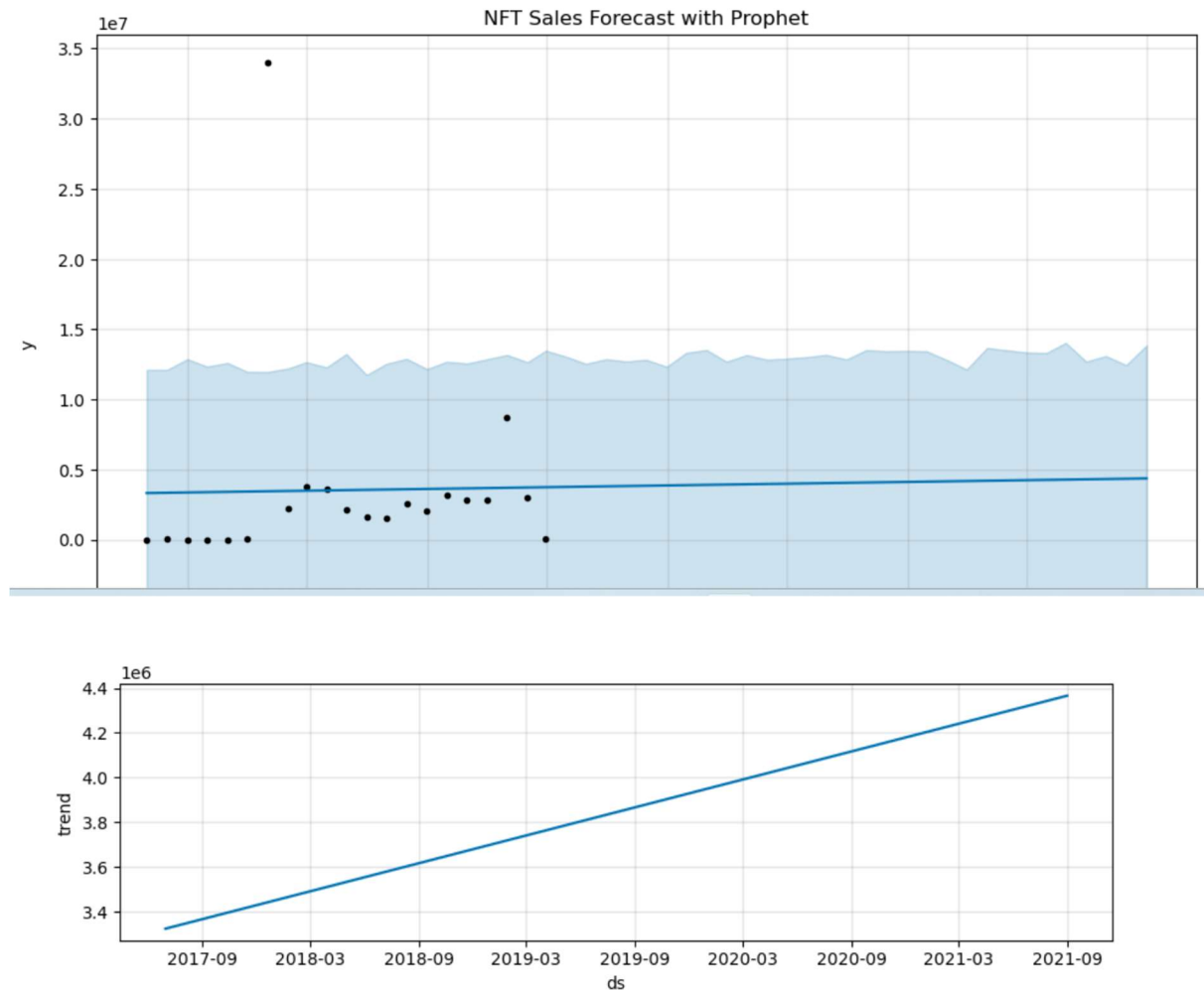
# Initialize and fit the Prophet model
model_prophet = Prophet()
model_prophet.fit(prophet_data)

# Forecast the next 30 months
future = model_prophet.make_future_dataframe(periods=30, freq='M')
forecast_prophet = model_prophet.predict(future)

# Plot the forecast
model_prophet.plot(forecast_prophet)
plt.title("NFT Sales Forecast with Prophet")
plt.show()

# Plot forecast components (trend, seasonality, etc.)
model_prophet.plot_components(forecast_prophet)
plt.show()

```



Evaluate Model Accuracy

We will evaluate the accuracy of both the ARIMA and Prophet models using appropriate metrics such as:

1. Mean Absolute Error (MAE): Measures the average magnitude of the errors in the predictions.
2. Root Mean Squared Error (RMSE): A measure of the differences between values predicted by a model and the actual values.

```

import numpy as np
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Use monthly_sales for actual vs predicted comparison
# Take only the last 21 months of actual data since it seems y_true has 21 months of data
y_true = monthly_sales[-21:] # Last 21 months actual sales

# Take the first 21 months of the ARIMA and Prophet forecasts to match the length of y_true
y_pred_arima = forecast_arima[:21] # ARIMA forecast (first 21 months)
y_pred_prophet = forecast_prophet.set_index('ds')['yhat'][-21:] # Prophet forecast (last 21 months)

# Calculate MAE and RMSE for ARIMA
mae_arima = mean_absolute_error(y_true, y_pred_arima)
rmse_arima = np.sqrt(mean_squared_error(y_true, y_pred_arima))

print(f"ARIMA Model - MAE: {mae_arima}, RMSE: {rmse_arima}")

# Calculate MAE and RMSE for Prophet
mae_prophet = mean_absolute_error(y_true, y_pred_prophet)
rmse_prophet = np.sqrt(mean_squared_error(y_true, y_pred_prophet))

print(f"Prophet Model - MAE: {mae_prophet}, RMSE: {rmse_prophet}")

ARIMA Model - MAE: 3930488.5482728155, RMSE: 7628949.947305241
Prophet Model - MAE: 3887868.7285934673, RMSE: 7108026.659824154

```

Key Insights:

- Prophet Model has a lower MAE and RMSE compared to the ARIMA model:
 - MAE: Prophet has a slightly lower MAE, indicating that the average difference between predicted and actual values is smaller in the Prophet model than in ARIMA.
 - RMSE: Prophet's RMSE is also lower, which means that the magnitude of larger errors (since RMSE penalizes larger errors more) is smaller compared to ARIMA.
- Based on the metrics, the Prophet model performs better than the ARIMA model for this dataset, with lower errors both in terms of MAE and RMSE. It suggests that Prophet may be more effective at capturing the underlying trends or seasonality in the data compared to ARIMA, at least for this particular use case.

To implement real-time data integration and automate the data collection process from NFT marketplaces, you need a system that continuously fetches and updates your dataset with new data as it becomes available.

Identify NFT Marketplaces with APIs

Most NFT marketplaces offer APIs that allow developers to retrieve real-time data on NFTs, such as sales, pricing, and activity. Popular NFT marketplaces with APIs include:

- OpenSea API: Provides real-time data on NFTs, collections, and marketplace activities.
- Setup Authentication and API Requests
- API Access: Most NFT marketplace APIs require authentication through API keys. Start by creating an account with the respective NFT marketplace and obtaining an API key.

Make API Calls to Fetch Real-Time Data: Here's an example of how to fetch data from OpenSea's API:

```

# Replace with your actual OpenSea API key
API_KEY = 'd3e02ba9f5de4d8099b0bb20915ef905'
headers = {"X-API-KEY": API_KEY}

# Define the API endpoint for fetching data
url = "https://api.opensea.io/api/v1/assets"

# Define parameters for the data request
params = {
    "order_direction": "desc",
    "offset": "0",
    "limit": "20", # Fetch 20 assets
    "collection": "boredapeyachtclub" # Example collection
}

# Fetch data
response = requests.get(url, headers=headers, params=params)

# Check if the request was successful
if response.status_code == 200:
    data = response.json()
    print("Data fetched successfully:")
    print(data)
else:
    print(f"Error fetching data: {response.status_code}")
    print(response.text) # Print the error message for debugging

```

Creating a Python-based dashboard to visualize NFT market trends using Plotly and Dash is an excellent way to display your real-time data. Below is a step-by-step guide to build a simple interactive dashboard that visualizes NFT market trends such as sales volume, average price, and active wallets.

Create the Python Script

Open a text editor like Notepad or VS Code.

```

File Edit View

# Initialize the Dash app
app = dash.Dash(__name__)

# Define the app layout
app.layout = html.Div(
    children=[
        html.H1("NFT Market Trends Dashboard", style={"textAlign": "center"}),

        # Dropdown for selecting different metrics
        html.Div([
            html.Label("Select Metric"),
            dcc.Dropdown(
                id="metric_dropdown",
                options=[
                    {"label": "Sales (USD)", "value": "Sales_USD"},
                    {"label": "Average Price (USD)", "value": "Average_Price"},
                    {"label": "Volume (USD)", "value": "Volume_USD"},
                    {"label": "Active Wallets", "value": "Active_Wallets"},
                ],
                value="Sales_USD", # Default value
                clearable=False,
            )
        ])
    ]
)

```

Once the file is saved, open the Command Prompt and run the following command: This should now run the Dash app, and you will be able to access it in your browser at <http://127.0.0.1:8050/>.

Data Integration

While this project initially used sample data, it is designed for easy integration with real-time data from NFT marketplaces like OpenSea. The plan is to fetch data via API calls and update the dashboard dynamically using the following:

- **API Integration:** Fetch data from sources like OpenSea using their API.
- **Real-Time Updates:** Schedule API calls or trigger them manually to refresh the dataset.



How the Analysis and Insights from the NFT Dashboard Could Benefit Artists, Collectors, and Investors in the NFT Ecosystem

The analysis and insights generated by the NFT dashboard can provide significant value to various participants in the NFT ecosystem, including artists, collectors, and investors. By visualizing key metrics such as sales volume, average price, and wallet activity, the dashboard allows these stakeholders to make data-driven decisions and better understand market dynamics.

- **Benefits for Artists**

Pricing Strategy

Artists can use the insights from the dashboard to establish pricing strategies based on real-time market trends. By analyzing average prices of NFTs within their genre or category, artists can:

- Set competitive prices for their artwork.
- Avoid under-pricing their NFTs in a rapidly evolving market.
- Track how their prices compare with other creators and adjust pricing accordingly to maximize revenue.

Market Timing

By studying the sales volume and transaction activity, artists can identify high-demand periods and time their NFT releases for optimal exposure and sales. For example:

- If the data shows a spike in active wallets or volume, artists can release new collections during those periods to take advantage of increased marketplace activity.
- This insight allows artists to maximize their reach and engagement by entering the market during peak demand periods.

Recognition of Trends and Popularity

The dashboard's ability to show market trends can help artists align their creative direction with current demands. By analyzing which categories of NFTs are currently performing well, artists can:

- Adjust their styles or themes based on the popularity of specific collections or genres.
- Tap into emerging trends, such as digital collectibles, generative art, or metaverse-related NFTs, by identifying which categories show a growth in volume and sales.

Community Growth and Targeting

The dashboard's data can help artists identify key wallet activity, including who is purchasing specific types of NFTs. Artists can engage with these buyers and communities to expand their reach, form collaborations, or host exclusive events.

- **Benefits for Collectors**

Smart Purchase Decisions

Collectors can leverage the dashboard's insights to make smarter, more informed purchasing decisions. The visualizations help collectors:

- Identify NFTs with rising sales volumes, indicating growing interest or popularity.
- Track price trends and decide whether to purchase or wait based on past and projected price movements.

By analyzing volume and average price data, collectors can also avoid buying into bubbles or overpaying for NFTs that may not hold their value.

Portfolio Diversification

Collectors looking to diversify their NFT portfolios can use the dashboard to identify emerging categories and under-the-radar NFT projects. For example:

- By monitoring active wallet growth in new or niche collections, collectors can discover opportunities before these collections go mainstream.
- The ability to visualize multiple metrics over time allows collectors to see potential areas for investment and diversification.

Timing Sales for Maximum Profit

In addition to purchasing, collectors looking to sell NFTs can use the dashboard to optimize their sales timing. For example:

- By tracking sales volume and average prices, collectors can sell NFTs when demand is high, maximizing profits.
- They can monitor fluctuations in the number of active wallets and transaction volume to sell when marketplace activity spikes.

Identifying Value Trends

By analyzing historical data on price trends and sales volume, collectors can recognize value patterns, such as which types of NFTs appreciate over time and which ones lose value. This helps collectors build long-term strategies for buying and holding NFTs.

- Benefits for Investors

Market Trend Analysis

Investors can use the dashboard to track macro-level trends in the NFT market. By analyzing metrics like sales volume and average prices, they can:

- Predict market cycles, including when to invest heavily in NFTs and when to wait.
- Identify market downturns or bubbles by monitoring drastic changes in volume or wallet activity.

For investors managing large portfolios, the dashboard can serve as a risk management tool, providing real-time indicators of potential shifts in market dynamics.

Identifying Emerging Opportunities

Investors can identify early-stage opportunities by tracking growth in active wallets and sales in specific categories or collections. As emerging artists and projects gain traction, early investment can lead to high returns.

- By comparing volume and price trends, investors can detect undervalued NFTs or projects with strong future potential.
- Historical price and volume trends help investors project future returns based on previous performance in similar markets.

Liquidity Tracking

Liquidity is essential for investors looking to buy and sell NFTs efficiently. The dashboard provides insights into market activity, showing where liquidity is highest. This helps investors ensure they are operating in liquid markets where their NFTs can be bought or sold quickly.

By visualizing changes in sales volume and transaction frequency, investors can:

- Plan their entry and exit strategies based on liquidity conditions.
- Identify which collections or categories are experiencing consistent high volumes, providing better opportunities for profitable trades.

Risk Management

NFT markets can be highly speculative, so risk management is crucial. Investors can use the dashboard to monitor:

- Sales volume volatility: High fluctuations in sales can indicate risky markets or speculative bubbles.
- Price trends: Tracking average price growth and declines helps investors avoid overpaying or selling too early.
- Market corrections: By analyzing historical price trends and transaction volumes, investors can anticipate corrections and plan accordingly.

Conclusion

This project successfully delivered a Python-based interactive dashboard that visualizes NFT market trends. Future work includes integrating live data from NFT marketplaces and deploying the dashboard for public use.

Attachments:

The full project code can be found in the GitHub Repository.

