Report: Data Analysis on Crop Prices

Author	**DDAMULIRA OWEN JAMES**
Contact email	**ddamuliraowen3@gmail.com**
Github link	**https://github.com/ddamulira-owen/Data-Analysis-on-Crop-prices**

1. Introduction

This report presents the findings from a project aimed at analyzing crop prices using data collected from the National Agricultural Statistics Service **(NASS) API**. The project involves developing a data pipeline that collects, processes, and analyzes agricultural data. The goal is to identify trends and patterns in crop prices, specifically focusing on corn prices in the United States from 2020 to 2023. The following sections detail the methodology, analysis, and insights gained from this data.

2. Data Collection

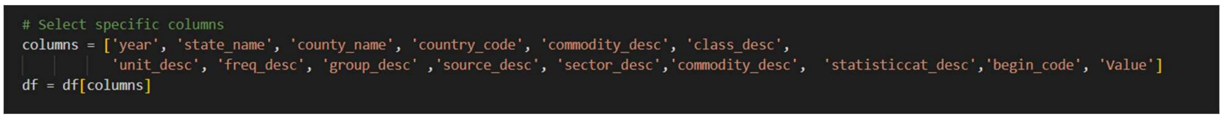
The data was collected using a Python script that interfaces with the NASS API. The script (**cropsnew.py**) retrieves corn price data for the years 2020 to 2023 and saves it into a CSV file named **crop_prices.csv**. The key steps in the data collection process include:

- **API Request:** The script makes HTTP GET requests to the NASS API using specific parameters such as the commodity type (corn), source description (survey), and the year.
- **Data Storage:** The retrieved data is stored in a Pandas Data Frame, filtered for relevant columns, and then concatenated into a single Data Frame. A sample of 200 rows is saved into a CSV file for analysis.

3. Data Processing

Data preprocessing involved cleaning and transforming the raw data to ensure it was suitable for analysis. The key steps included:

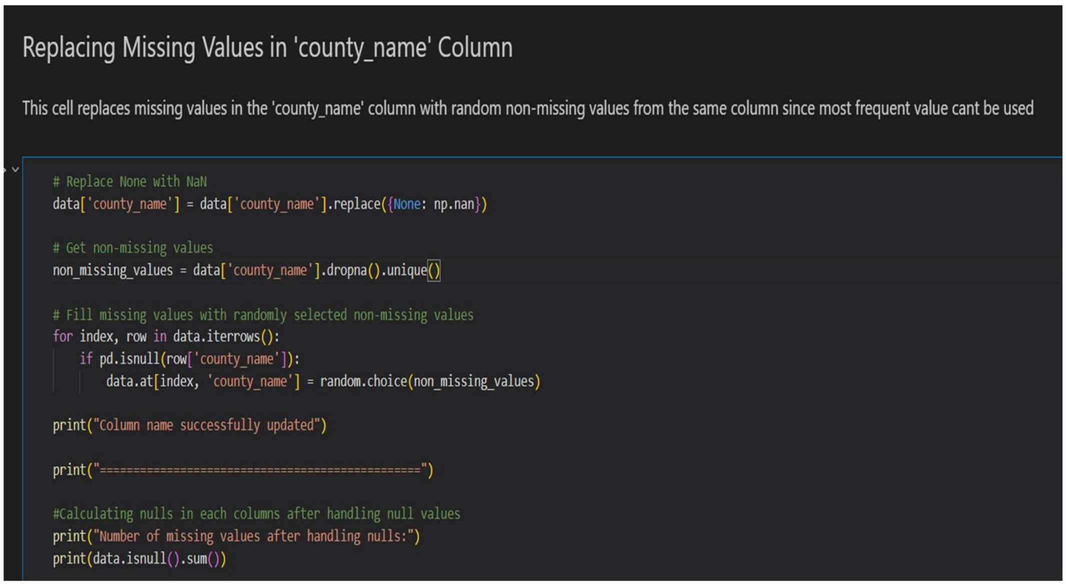
Filtering Columns: Selecting relevant columns such as year, state, county, commodity description, unit description, and price value.



```
# Select specific columns
columns = ['year', 'state_name', 'county_name', 'country_code', 'commodity_desc', 'class_desc',
           'unit_desc', 'freq_desc', 'group_desc' ,'source_desc', 'sector_desc','commodity_desc',  'statisticcat_desc','begin_code', 'Value']
df = df[columns]
```

Handling Missing Values:

- The missing values in the 'county_name' column were replaced with random non-missing values.

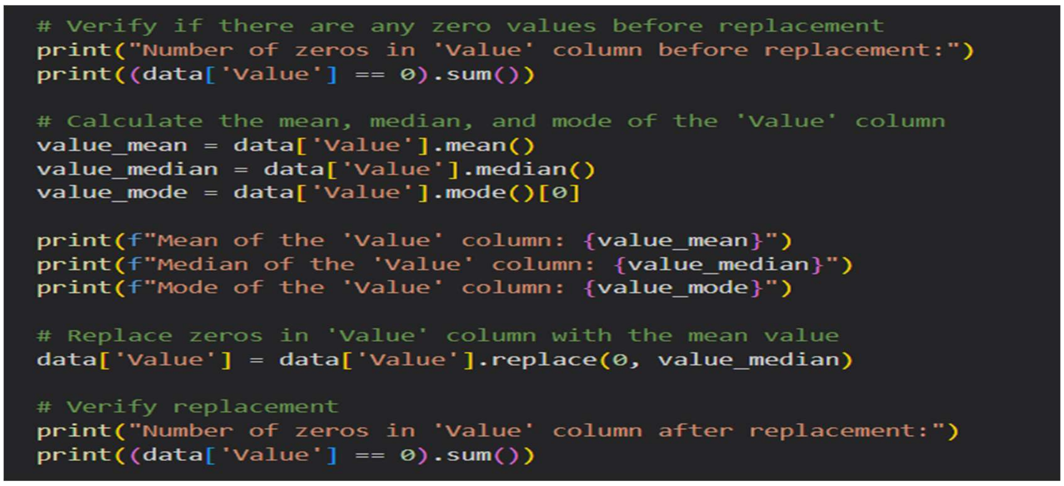


Replacing Missing Values in 'county_name' Column

This cell replaces missing values in the 'county_name' column with random non-missing values from the same column since most frequent value cant be used

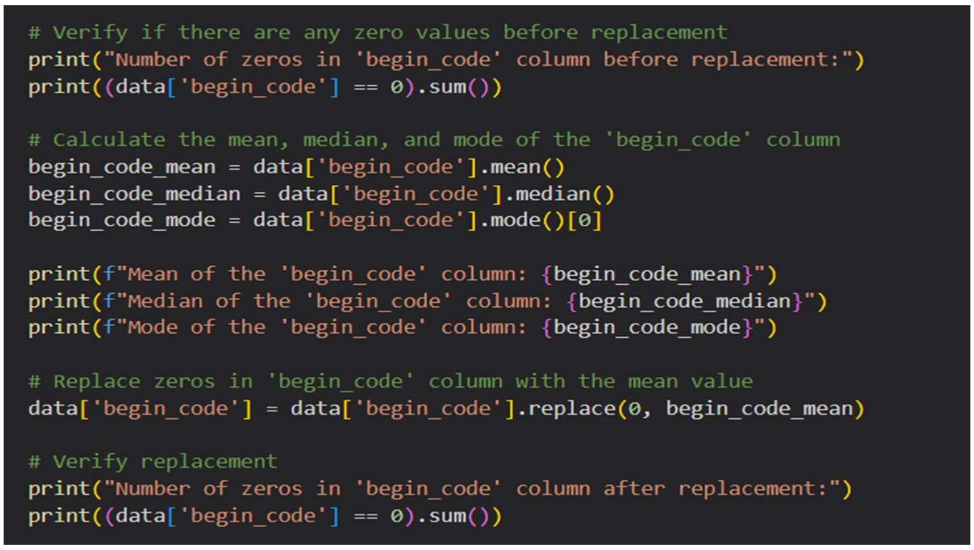
```python
# Replace None with NaN
data['county_name'] = data['county_name'].replace({None: np.nan})

# Get non-missing values
non_missing_values = data['county_name'].dropna().unique()

# Fill missing values with randomly selected non-missing values
for index, row in data.iterrows():
    if pd.isnull(row['county_name']):
        data.at[index, 'county_name'] = random.choice(non_missing_values)

print("Column name successfully updated")

print("================================================")

#Calculating nulls in each columns after handling null values
print("Number of missing values after handling nulls:")
print(data.isnull().sum())
```

- Rows with missing or non-integer values in the 'Value' column were removed.
- Zeros in the 'Value' column were replaced with the median value to maintain data integrity.



```python
# Verify if there are any zero values before replacement
print("Number of zeros in 'Value' column before replacement:")
print((data['Value'] == 0).sum())

# Calculate the mean, median, and mode of the 'Value' column
value_mean = data['Value'].mean()
value_median = data['Value'].median()
value_mode = data['Value'].mode()[0]

print(f"Mean of the 'Value' column: {value_mean}")
print(f"Median of the 'Value' column: {value_median}")
print(f"Mode of the 'Value' column: {value_mode}")

# Replace zeros in 'Value' column with the mean value
data['Value'] = data['Value'].replace(0, value_median)

# Verify replacement
print("Number of zeros in 'Value' column after replacement:")
print((data['Value'] == 0).sum())
```

- Zeros in the 'begin_code' column were replaced with the median value to maintain data integrity

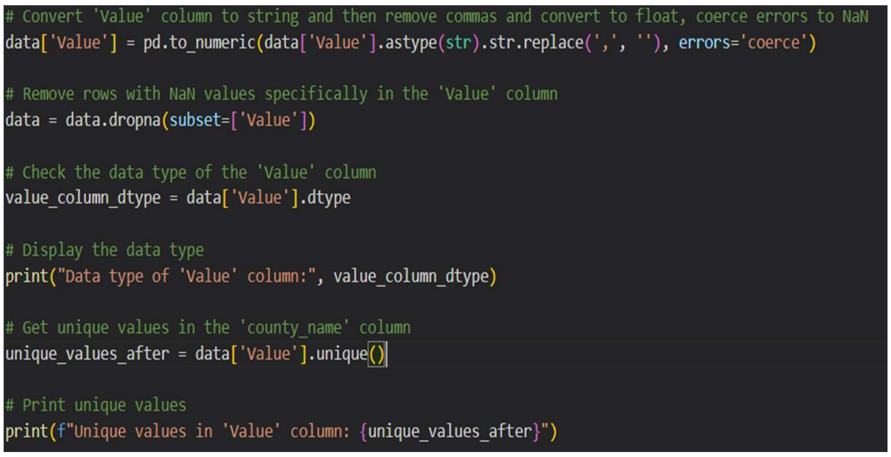


```python
# Verify if there are any zero values before replacement
print("Number of zeros in 'begin_code' column before replacement:")
print((data['begin_code'] == 0).sum())

# Calculate the mean, median, and mode of the 'begin_code' column
begin_code_mean = data['begin_code'].mean()
begin_code_median = data['begin_code'].median()
begin_code_mode = data['begin_code'].mode()[0]

print(f"Mean of the 'begin_code' column: {begin_code_mean}")
print(f"Median of the 'begin_code' column: {begin_code_median}")
print(f"Mode of the 'begin_code' column: {begin_code_mode}")

# Replace zeros in 'begin_code' column with the mean value
data['begin_code'] = data['begin_code'].replace(0, begin_code_mean)

# Verify replacement
print("Number of zeros in 'begin_code' column after replacement:")
print((data['begin_code'] == 0).sum())
```

➢ Columns with a single repeated value were removed as they do not add meaningful information.

Data Cleaning:

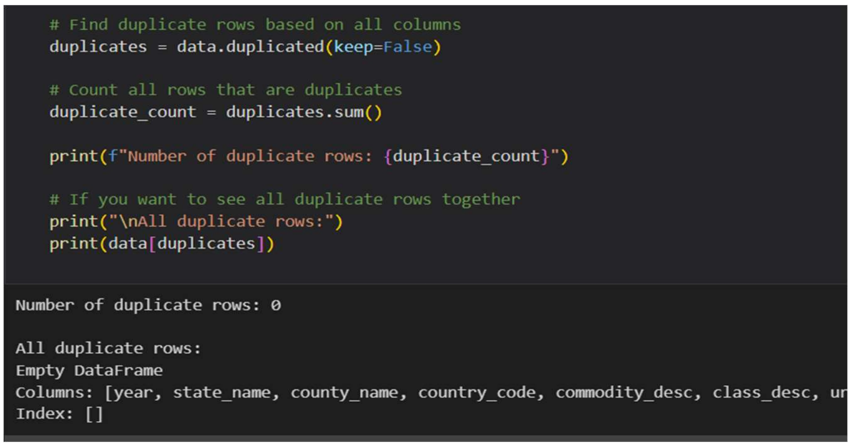
➢ Commas were removed from numeric columns, and values were converted to float.



```python
# Convert 'Value' column to string and then remove commas and convert to float, coerce errors to NaN
data['Value'] = pd.to_numeric(data['Value'].astype(str).str.replace(',', ''), errors='coerce')

# Remove rows with NaN values specifically in the 'Value' column
data = data.dropna(subset=['Value'])

# Check the data type of the 'Value' column
value_column_dtype = data['Value'].dtype

# Display the data type
print("Data type of 'Value' column:", value_column_dtype)

# Get unique values in the 'county_name' column
unique_values_after = data['Value'].unique()

# Print unique values
print(f"Unique values in 'Value' column: {unique_values_after}")
```

➢ Incorrect data types were corrected, ensuring consistency across the dataset.

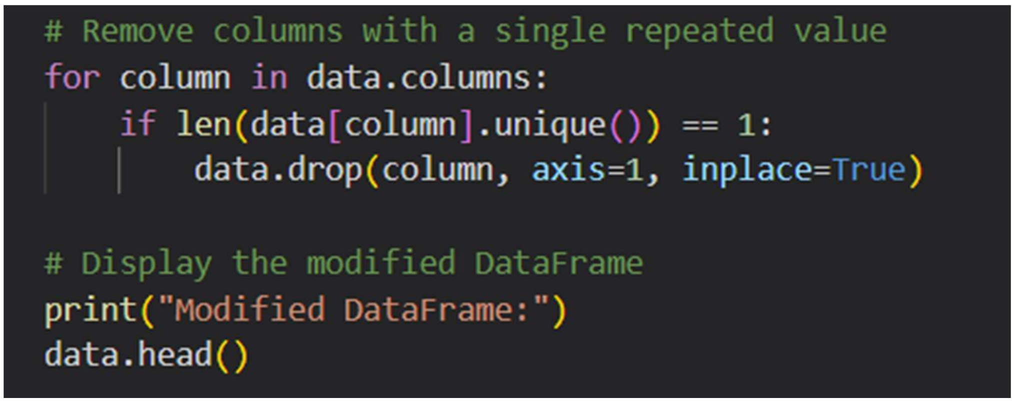
➢ No duplicates found in the data



```
# Find duplicate rows based on all columns
duplicates = data.duplicated(keep=False)

# Count all rows that are duplicates
duplicate_count = duplicates.sum()

print(f"Number of duplicate rows: {duplicate_count}")

# If you want to see all duplicate rows together
print("\nAll duplicate rows:")
print(data[duplicates])
```

```
Number of duplicate rows: 0

All duplicate rows:
Empty DataFrame
Columns: [year, state_name, county_name, country_code, commodity_desc, class_desc, ur
Index: []
```

- Identifying and removing columns with a single repeated value.



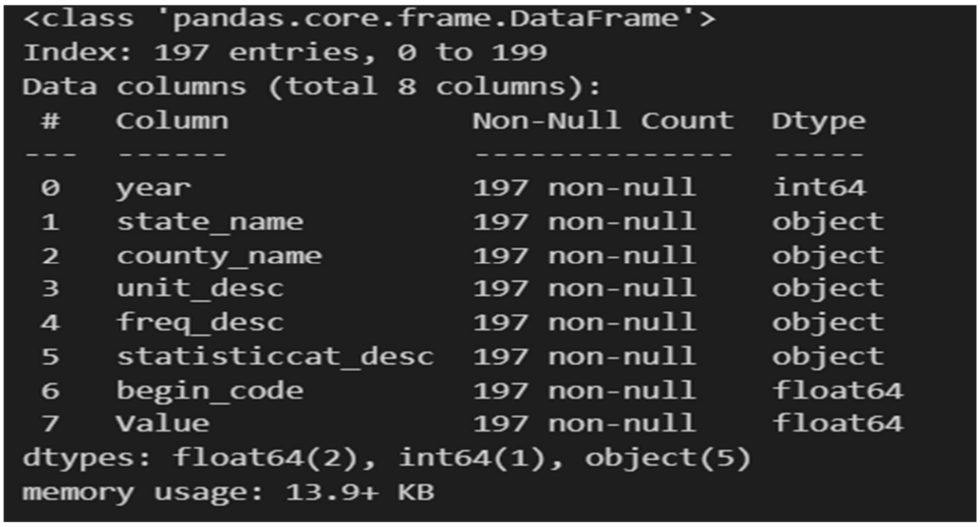
```
# Remove columns with a single repeated value
for column in data.columns:
    if len(data[column].unique()) == 1:
        data.drop(column, axis=1, inplace=True)

# Display the modified DataFrame
print("Modified DataFrame:")
data.head()
```

4. Data Analysis

Following data cleaning, various exploratory techniques were applied to understand the data's characteristics:

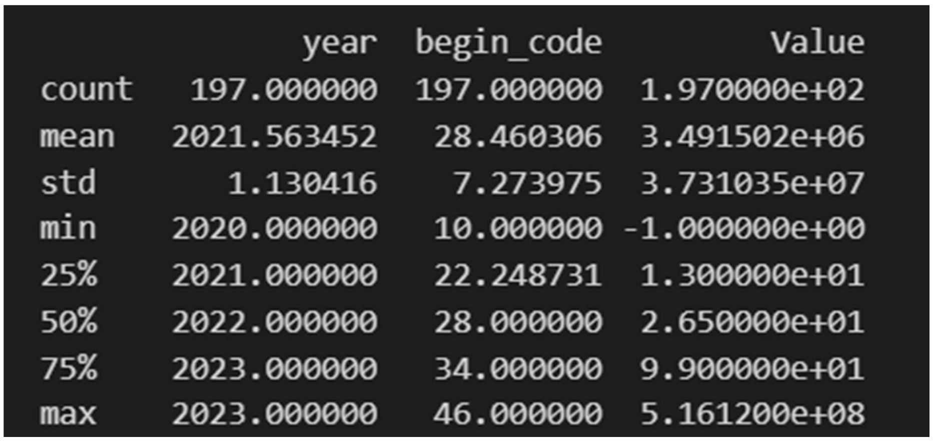
Unique Values: We examined the unique values in each column to identify potential anomalies or inconsistencies.

Data Types: We verified the data types of each column to ensure proper manipulation and analysis.



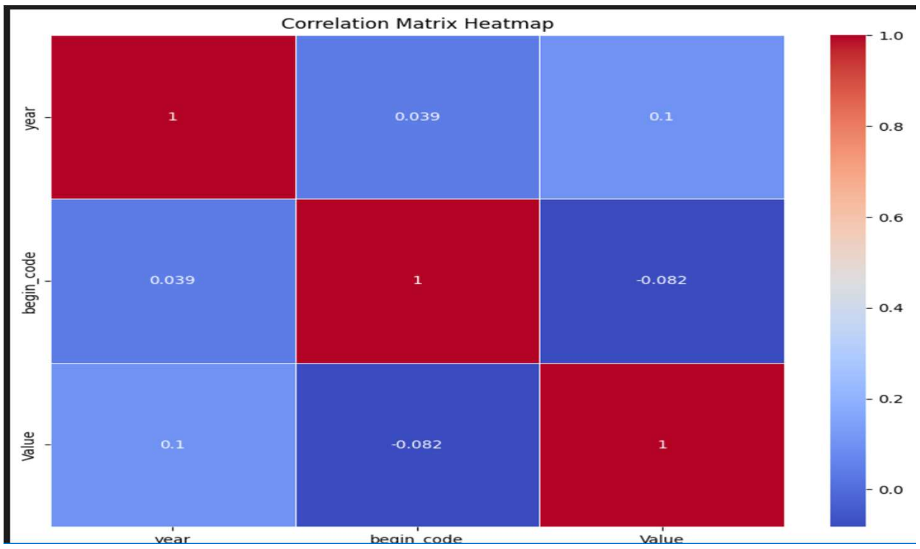
```
<class 'pandas.core.frame.DataFrame'>
Index: 197 entries, 0 to 199
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   year              197 non-null    int64  
 1   state_name        197 non-null    object 
 2   county_name       197 non-null    object 
 3   unit_desc         197 non-null    object 
 4   freq_desc         197 non-null    object 
 5   statisticcat_desc 197 non-null    object 
 6   begin_code        197 non-null    float64
 7   Value             197 non-null    float64
dtypes: float64(2), int64(1), object(5)
memory usage: 13.9+ KB
```

Summary Statistics: Descriptive statistics like mean, median, standard deviation, minimum, and maximum values were calculated for numeric columns to summarize the central tendency and spread of the data.

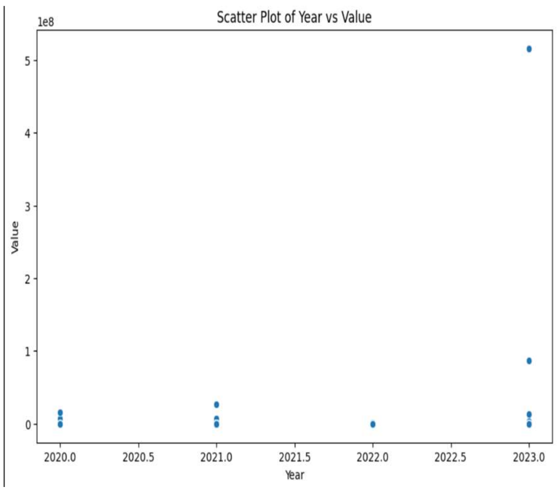


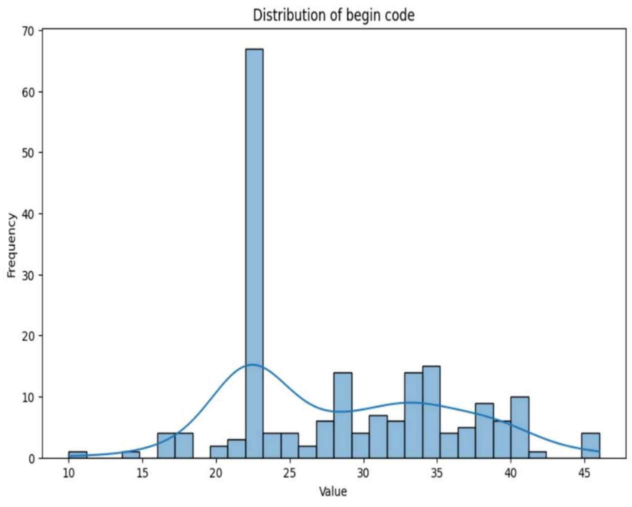
```
              year  begin_code         Value
count   197.000000  197.000000  1.970000e+02
mean   2021.563452   28.460306  3.491502e+06
std       1.130416    7.273975  3.731035e+07
min    2020.000000   10.000000 -1.000000e+00
25%    2021.000000   22.248731  1.300000e+01
50%    2022.000000   28.000000  2.650000e+01
75%    2023.000000   34.000000  9.900000e+01
max    2023.000000   46.000000  5.161200e+08
```

Correlation matrix and heat map were generated to identify relationships between numeric features.

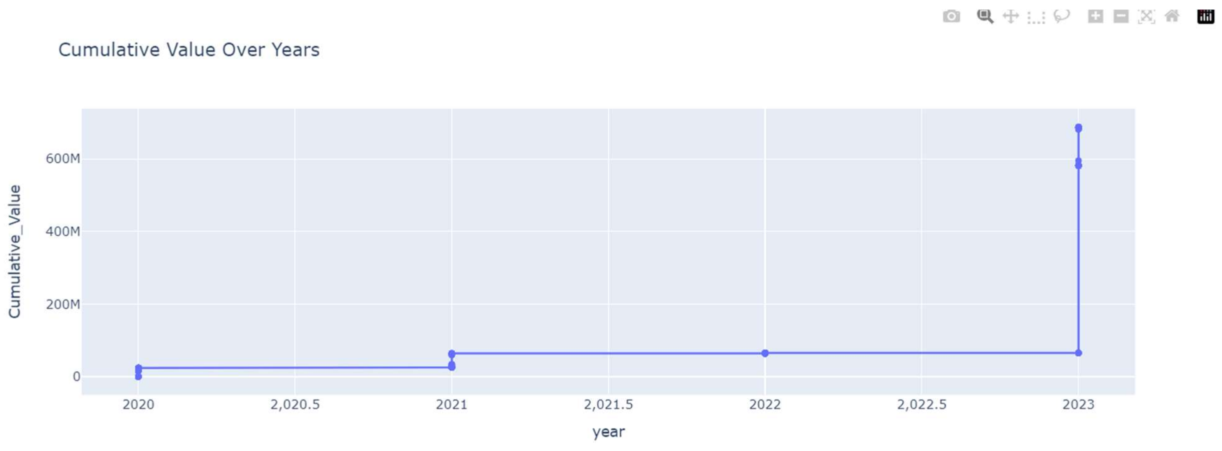


Histograms and scatter plots were created to visualize the distribution of values and trends over time.

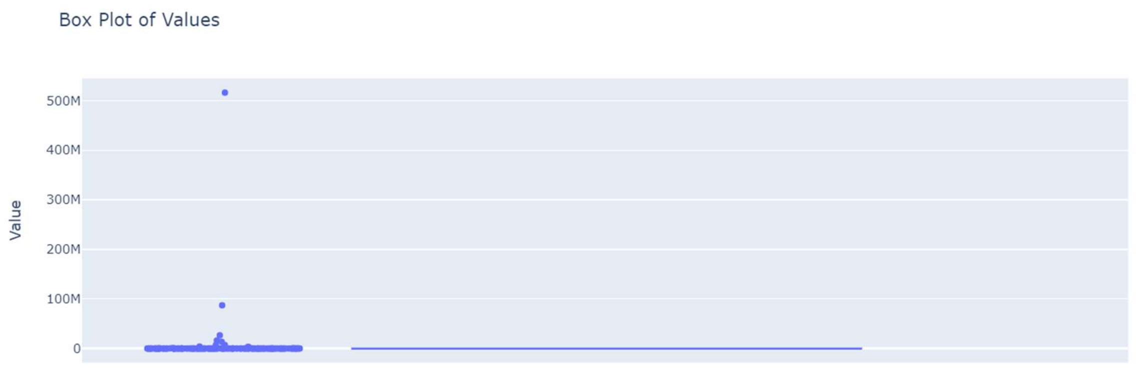




Cumulative Value: A cumulative line graph was generated to show the total value of crops over the years. This helps understand how the overall value of crop production has changed over time.



Price Distribution by State. The box plot highlights the distribution of corn prices across different states. It is evident that certain states exhibit higher price variability, which could be attributed to regional agricultural practices and climatic conditions.



5. Findings and Recommendations

Key Insights

- The dataset contains information about crop prices, with variations observed over the years.
- The 'Value' column exhibits a right-skewed distribution, indicating potential outliers or significant differences in crop prices.
- There is a moderate positive correlation between certain numeric features, suggesting some level of dependency.

- The cumulative analysis shows an increasing trend in crop values over the years, with fluctuations in certain periods.

Actionable Recommendations

- **Further Data Exploration:** Conduct deeper analysis to understand the factors influencing crop prices, such as market trends, agricultural practices, and external factors like weather conditions.
- **Outlier Detection and Handling:** Identify and investigate potential outliers in the 'Value' column to determine if they are valid data points or errors. Outliers could provide insights into exceptional crop price fluctuations.
- **Time Series Analysis:** Perform time series analysis to forecast future crop prices and identify seasonal patterns or long-term trends.
- **Enhanced Data Collection:** Collect additional data attributes such as geographical information, crop types, and market dynamics to enrich the analysis and provide more comprehensive insights.

6. Conclusion

This project successfully developed a data pipeline to collect, process, and analyze agricultural data from the NASS API. The analysis provided valuable insights into corn price trends and regional variations. The findings can inform agricultural policies and strategies to support farmers and enhance crop price stability.

7. Appendices

7.1. Code

The code can be found on my github page with this link.

https://github.com/ddamulira-owen/Data-Analysis-on-Crop-prices

8. References

National Agricultural Statistics Service (NASS) API Documentation: NASS Quick Stats

Python Data Analysis and Visualization Libraries: Pandas, Matplotlib, Seaborn

9. Acknowledgments

I would like to thank Raining Vegetables for the opportunity to work on this project and gain valuable experience in both software development and data analysis. Special thanks to the team for their support and guidance throughout the project.