Advanced-Data Analysis in Asset Management and Investment Strategies Using Python and Power BI.

1. INTRODUCTION OF THE COURSE

Advanced Data Analytics is a data analysis methodology using predictive modeling, machine learning algorithms, deep learning, business process automation, and other statistical methods to analyze business information from various data sources.

In today's data-driven world, analyzing and interpreting large volumes of data is essential for making informed decisions. In this course, "Advanced Data Analysis," I explored the skills and knowledge required to conduct advanced data analysis. Regardless of whether the data is structured or unstructured, this course aims to maximize the potential expertise of our data.

The course comprises a range of advanced methodologies and tools utilized in data analysis, encompassing statistical approaches, machine learning algorithms, and data visualization techniques. Which includes the processes of data cleansing, manipulation, and modeling, intending to reveal latent patterns, trends, and insights that can inform strategic decisionmaking.

In enriching this intellectual journey the goal is to adeptly transform raw data into actionable intelligence, enabling us to make informed decisions and drive meaningful outcomes. The use of programming languages like Python, and R and various visualization tools like Power BI and Tableau for interactive visualizations.

2. TECHNICAL LEARNINGS FROM THE COURSE

A. DATA/WEB SCRAPING

Data scraping is a technique used to extract large amounts of data from websites or other online sources. It involves systematically gathering and extracting relevant financial data from various online platforms to analyze asset management and investment trends. This process ensures that comprehensive and up-todate information is collected, which is essential for performing accurate and meaningful data analysis.

The data scraping process was performed through the following steps;

a) Studied the structure of the targeted websites to understand how the data is presented. This involves inspecting HTML elements to locate the data points you need.

b) Chose and configured a web scraping tool or library (such as BeautifulSoup, Scrapy, or Selenium) that is suitable for extracting data from the identified sources.

c) Developed a script that navigates through the website, accesses the relevant pages, and extracts the required data. This script will parse HTML content and identify the elements that contain the data.

d) If the data spans multiple pages, incorporate logic to handle pagination, ensuring that the script can navigate through all pages to collect comprehensive data.

e) **Tools and Libraries Used:**
   a. Python for scripting.
   b. BeautifulSoup and requests libraries for web scraping.
   c. Pandas for data manipulation and storage.
   d. Selenium for automated interactions with web browsers.
   e. Parsel extracting data from HTML and XML documents.
   f. Jupyter Notebook as the environment for the programming process.

f) **Target Website:** URL: tamis.co.tz/fund-performance

A. PRINCIPAL COMPONENT ANALYSIS

Principal Component Analysis (PCA) is a statistical technique used to simplify the complexity of high-dimensional data while retaining its most important information. When dealing with data that has many variables, it can be challenging to analyze and visualize the relationships between these variables. PCA helps by reducing the number of dimensions (or variables) without losing significant data insights.

Principal Component Analysis works as follows;

- Standardization: PCA starts by standardizing the data if the variables are on different scales. This ensures that each variable contributes equally to the analysis.
- Covariance Matrix Calculation: It then computes the covariance matrix, which shows how the variables in the dataset are correlated with each other. This step is crucial for understanding the relationships between variables.
- Eigenvectors and Eigenvalues: The next step involves calculating the eigenvectors and eigenvalues from the covariance matrix. Eigenvectors represent the directions of the maximum variance in the data, while eigenvalues show the magnitude of variance in these directions.
- Principal Components: The eigenvectors with the highest eigenvalues are selected as the principal components. These components are essentially new variables that are linear combinations of the original variables but capture the most significant patterns in the data.
- Dimensionality Reduction: By projecting the original data onto these principal components, PCA reduces the dataset's dimensions while retaining the most important information. This reduced dataset is easier to analyze and visualize.

C. POWER BI VISUALIZATION TOOL

Power BI is a business analytics tool developed by Microsoft for visualizing and sharing insights from data. It helps in preparing and visualizing data to communicate insights effectively. It is widely used in data analysis to create interactive reports and dashboards that help make data-driven decisions.

What the tool does;

- Power BI can connect to various data sources, making it easy to import data for analysis from different systems.
- Power BI provides tools to clean, transform, and structure data before visualization, ensuring it's ready for analysis tasks.
- Power BI offers a variety of interactive visualization options, including charts, graphs, maps, and tables, allowing for in-depth data exploration.
- Users can create comprehensive dashboards by combining multiple visualizations, providing a realtime view of key metrics.
- Power BI enables easy sharing of reports and dashboards with role-based access control within the organization, and the ability to publish reports to the web or embed them in other applications.
- Power BI includes AI capabilities to automatically generate insights, identify trends, and suggest visualizations based on the data, aiding in uncovering hidden patterns.

3.1 INTRODUCTION OF MINI PROJECT

The project focuses on using data analysis and predictive modeling to enhance decision-making for investors and fund managers. It aims to utilize data scraping, data integration, exploratory data analysis (EDA), and advanced predictive methods to gain a deeper understanding of market behaviors and asset performance.

The focus is on analyzing risks and potential returns to help stakeholders build investment portfolios that match their financial goals. Additionally, the project looks at analyzing market trends and predicting potential future developments to provide valuable insights and recommendations to investors, fund managers, and stakeholders.

OBJECTIVES

The project was performed to attain meaningful insights and recommendations that align with changing market conditions and investment goals. The following were the objectives that helped to achieve the goals;

i. Risk and Return Analysis:
- Assessing risk and return profiles of assets and funds and using quantitative methods to measure volatility and other risk metrics.

ii. Optimization of Investment Portfolios:
- Applying optimization techniques to create portfolios that maximize returns and minimize risks, while also incorporating constraints and investor preferences

iii. Market Trend Analysis
- Analyzing market trends and providing forecasts and scenario analyses to support strategic planning and decision-making.

i. Automation and Real-Time Reporting:
- Develop automated processes for data collection, analysis, and reporting, and create a dynamic Power BI dashboard for real-time monitoring of investment metrics.

ii. Stakeholder Insights and Recommendations:

- Translated analytical findings into actionable recommendations and presented insights using visualizations and summaries.

SCOPE OF THE PROJECT

The primary aim of this study is to conduct a comprehensive quantitative analysis of asset management and investment strategies, utilizing advanced data science techniques to enhance decision-making processes. The study focuses on evaluating the performance of various funds, identifying market trends, and optimizing investment portfolios to achieve a balanced risk-return profile.

The study analyzes data from a diverse range of funds managed by UTTAMIS. This includes detailed financial and performance data of various investment schemes, such as the Umoja Fund, Wekeza Maisha Fund, Watoto Fund, Jikimu Fund, Liquid Fund, and Bond Fund. The sample encompasses historical and current data points for these funds.

The study covers historical data ranging from the inception of each fund to the most recent available data. The analytical phase of the study is expected to take approximately six months, including data collection, analysis, model development, and reporting.

3.2 TECHNICAL ASSIGNMENTS

Throughout this data analysis course, I've gained a wealth of technical knowledge that has significantly enhanced my ability to work with data. Here are the key technical learnings I've taken away:

a) **Data Cleaning and Preprocessing**

- I learned the importance of preparing data before analysis. This includes handling missing values, filtering out irrelevant data, and ensuring consistency in the dataset. Proper data cleaning is the foundation of any successful analysis, and I've become proficient in using various tools and techniques to clean and preprocess data efficiently.

b) **Statistical Analysis**

- I've deepened my understanding of statistical methods and how they apply to data analysis. From descriptive statistics to more complex inferential techniques, I can now confidently use statistical tools to summarize data, identify trends, and draw meaningful conclusions.

c) **Machine Learning Algorithms**

- I explored the application of machine learning algorithms to real-world data problems. Whether it's regression, classification, or clustering, I now have the skills to select, implement, and fine-tune models to achieve accurate predictions and insights. d) **Data Visualization**

- Creating clear and impactful visualizations has been a major focus of my learning. I've mastered various visualization tools and techniques to present data in a way that is both understandable and compelling. Whether it's through charts, graphs, or dashboards, I can effectively communicate my findings to different audiences.

e) **Advanced Analytics Techniques**

- The course introduced me to advanced analytical methods like Principal Component Analysis (PCA) and time-series analysis. These techniques allow me to uncover deeper insights from complex datasets, reduce dimensionality, and analyze data trends over time.

f) **Use of Analytical Tools**

- I've gained hands-on experience with several analytical tools such as Python, R, Power BI, and Excel. Each tool has its strengths, and I now know how to leverage them depending on the task at hand, whether it's for data manipulation, model building, or visualization.

4. DETAILS OF MINI PROJECT

The main aim of the project was to develop a powerful data analysis tool that could pull out, organize, and display financial performance data from a particular website. The ultimate goal was to automate the process of extracting fund performance data from the TAMIS website, clean and process the data for analysis, and then present the results using advanced visualization tools.

PROCEDURE

1. Gathering and cleaning financial data using various techniques for analysis purposes.
2. Extracting, transforming, and loading scraped data for analysis and prediction.
3. Using statistical methods to uncover insights from the data in a preliminary analysis.
4. Applying machine learning algorithms to forecast how assets will perform.
5. Creating automated workflows and dynamic dashboards for real-time monitoring and reporting.
6. Generating actionable insights tailored for different stakeholder groups.

IMPORTANCE AND APPLICABILITY

This project is crucial for enhancing the way we understand and manage investments. By diving deep into the data from various funds, we can unearth valuable insights that help make smarter investment decisions. It's more about transforming complex data into clear, actionable strategies that investors can use to maximize their returns.

The findings from this study will benefit new and seasoned investors alike. New investors will gain a clearer picture of the potential risks and rewards, helping them make informed decisions immediately. Experienced investors and fund managers can use the insights to fine-tune their strategies, stay ahead of market trends, and ensure their portfolios are optimized for the best possible performance.

Moreover, this project isn't confined to just theoretical knowledge. Its real-world applicability means that the strategies and models developed here can be directly implemented to improve the efficiency and effectiveness of investment management. By automating data analysis and creating dynamic dashboards, we can ensure that decision-makers have access to up-to-date information, enabling them to react swiftly to market changes.

METHODOLOGY

C. DATA SCRAPING

Data scraping is a technique used to extract large amounts of data from websites or other online sources. It involves systematically gathering and extracting relevant financial data from various online platforms to analyze asset management and investment trends. This process ensures that comprehensive and up-todate information is collected, which is essential for performing accurate and meaningful data analysis.

The data scraping process was performed through the following steps;

g) Studied the structure of the targeted websites to understand how the data is presented. This involves inspecting HTML elements to locate the data points you need.

h) Chose and configured a web scraping tool or library (such as BeautifulSoup, Scrapy, or Selenium) that is suitable for extracting data from the identified sources.

i) Developed a script that navigates through the website, accesses the relevant pages, and extracts the required data. This script will parse HTML content and identify the elements that contain the data.

j) If the data spans multiple pages, incorporate logic to handle pagination, ensuring that the script can navigate through all pages to collect comprehensive data.

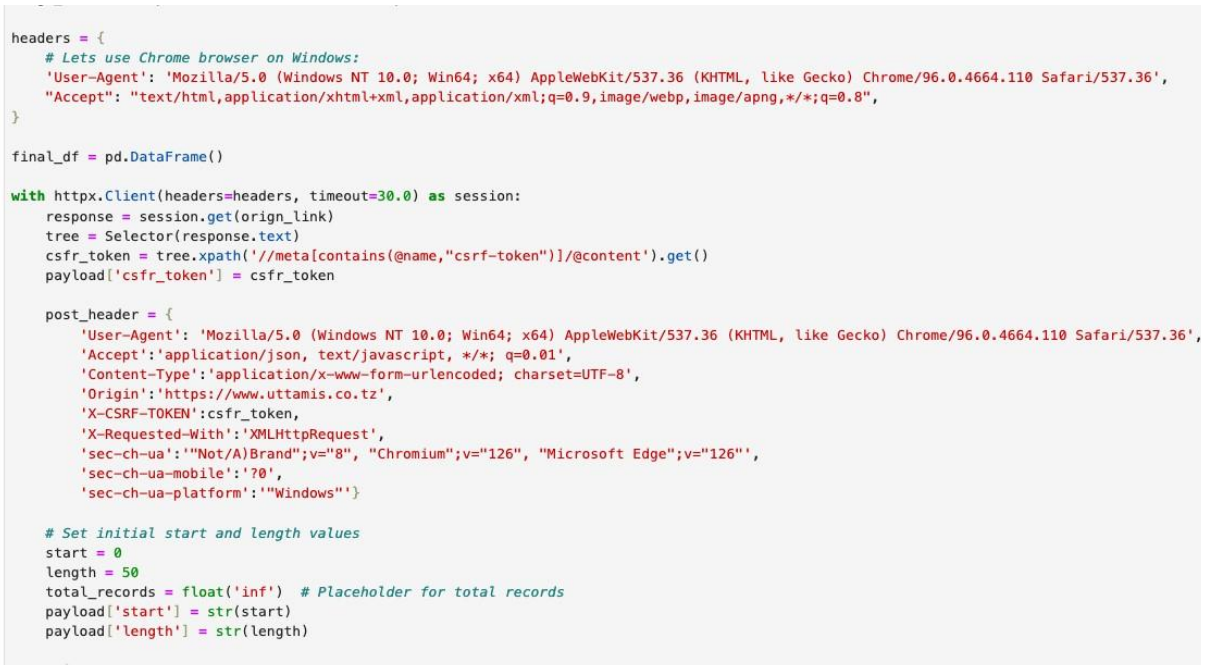
k) **Tools and Libraries Used:**

   a. Python for scripting.
   b. BeautifulSoup and requests libraries for web scraping.
   c. Pandas for data manipulation and storage.
   d. Selenium for automated interactions with web browsers.
   e. Parsel extracting data from HTML and XML documents.
   f. Jupyter Notebook as the environment for the programming process.

l) **Target Website:** URL: tamis.co.tz/fund-performance

m) **Code Snippets**

• Here are the key code snippets used in the data scraping process:

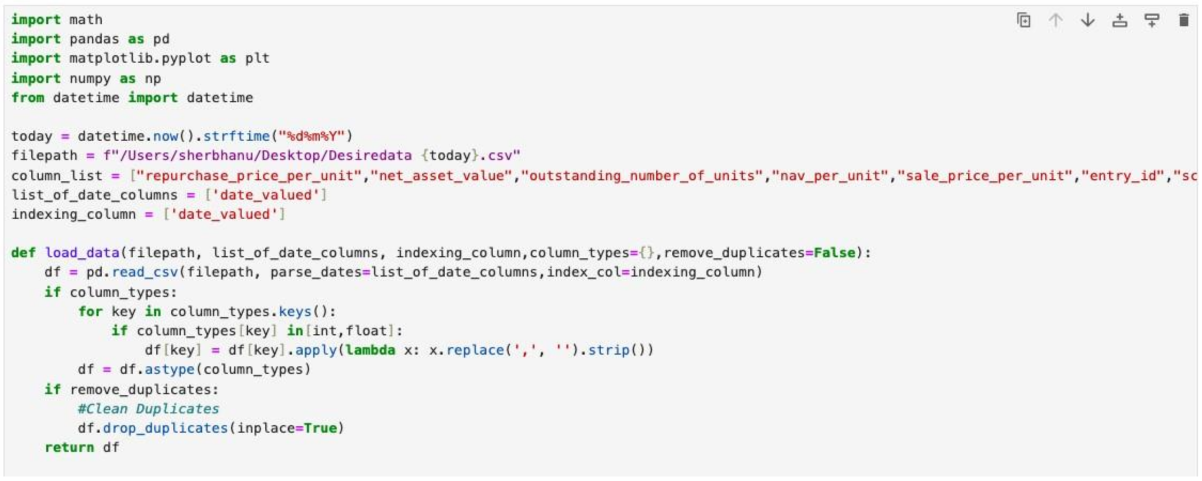


```
headers = {
    # Lets use Chrome browser on Windows:
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.110 Safari/537.36',
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8",
}

final_df = pd.DataFrame()

with httpx.Client(headers=headers, timeout=30.0) as session:
    response = session.get(orign_link)
    tree = Selector(response.text)
    csfr_token = tree.xpath('//meta[contains(@name,"csrf-token")]/@content').get()
    payload['csfr_token'] = csfr_token

    post_header = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.110 Safari/537.36',
        'Accept':'application/json, text/javascript, */*; q=0.01',
        'Content-Type':'application/x-www-form-urlencoded; charset=UTF-8',
        'Origin':'https://www.uttamis.co.tz',
        'X-CSRF-TOKEN':csfr_token,
        'X-Requested-With':'XMLHttpRequest',
        'sec-ch-ua':'"Not/A)Brand";v="8", "Chromium";v="126", "Microsoft Edge";v="126"',
        'sec-ch-ua-mobile':'?0',
        'sec-ch-ua-platform':'"Windows"'}

    # Set initial start and length values
    start = 0
    length = 50
    total_records = float('inf')  # Placeholder for total records
    payload['start'] = str(start)
    payload['length'] = str(length)
```

Code Snippet 1: Data Scraping Process of the desired data

n) **Data Collection Process**

- Data Extraction:
  - The script sends an HTTP GET request to the target URL to fetch the HTML content. ○ Parsel is used to parse the HTML and locate the specific fund performance data table.



```python
import math
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from datetime import datetime

today = datetime.now().strftime("%d%m%Y")
filepath = f"/Users/sherbhanu/Desktop/Desiredata {today}.csv"
column_list = ["repurchase_price_per_unit","net_asset_value","outstanding_number_of_units","nav_per_unit","sale_price_per_unit","entry_id","sc
list_of_date_columns = ['date_valued']
indexing_column = ['date_valued']

def load_data(filepath, list_of_date_columns, indexing_column,column_types={},remove_duplicates=False):
    df = pd.read_csv(filepath, parse_dates=list_of_date_columns,index_col=indexing_column)
    if column_types:
        for key in column_types.keys():
            if column_types[key] in[int,float]:
                df[key] = df[key].apply(lambda x: x.replace(',', '').strip())
        df = df.astype(column_types)
    if remove_duplicates:
        #Clean Duplicates
        df.drop_duplicates(inplace=True)
    return df
```

Code Snippet 2: Extraction, Transformation, and Loading of the Dataset

- Data Parsing:
    - The headers and rows of the table are extracted and stored in lists. ○ These lists are then used to create a Pandas DataFrame, which provides a structured format for the data.
- Data Storage:
    - The DataFrame is saved to a CSV file named 'FundData.csv' for easy access and analysis.

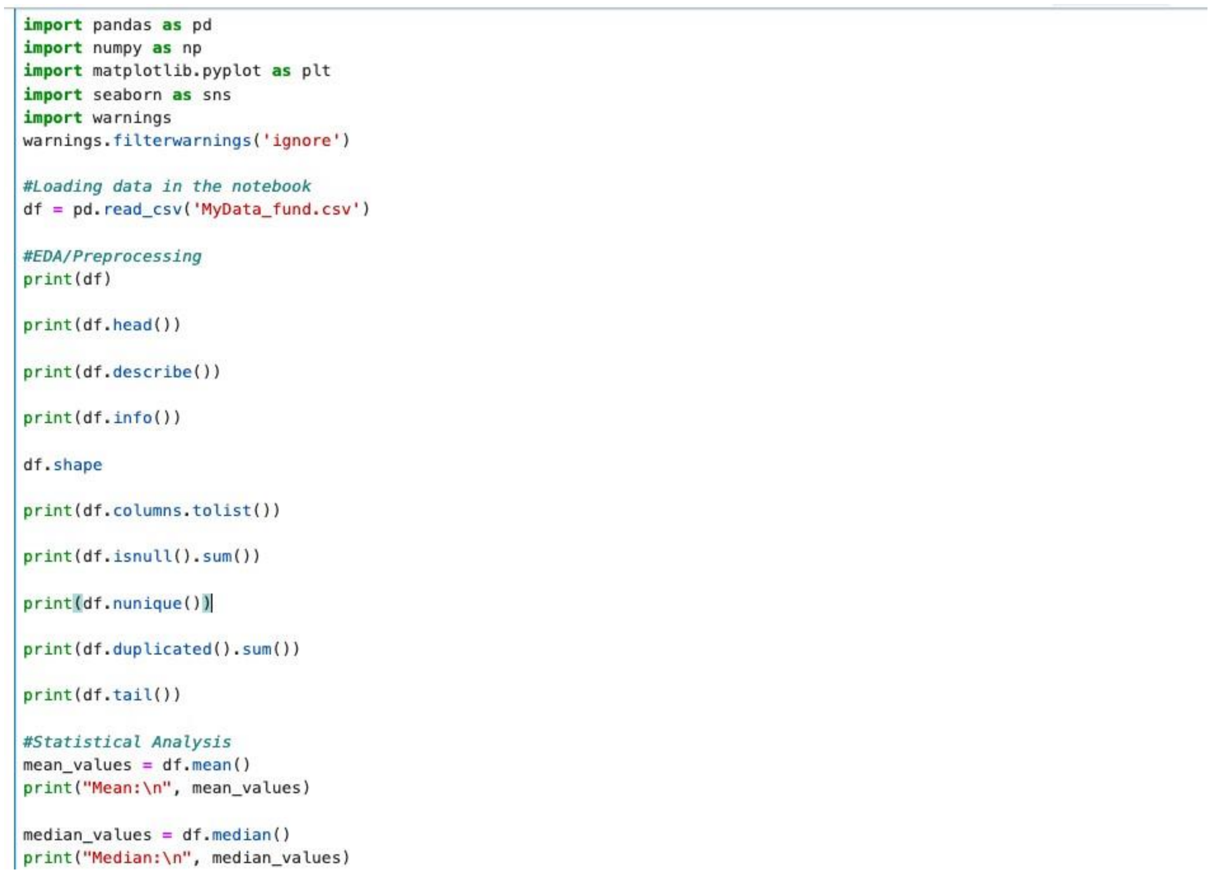
B: EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis (EDA) is a crucial initial step in data analysis, aimed at understanding the underlying structure and patterns within a dataset. For the project, EDA involves:

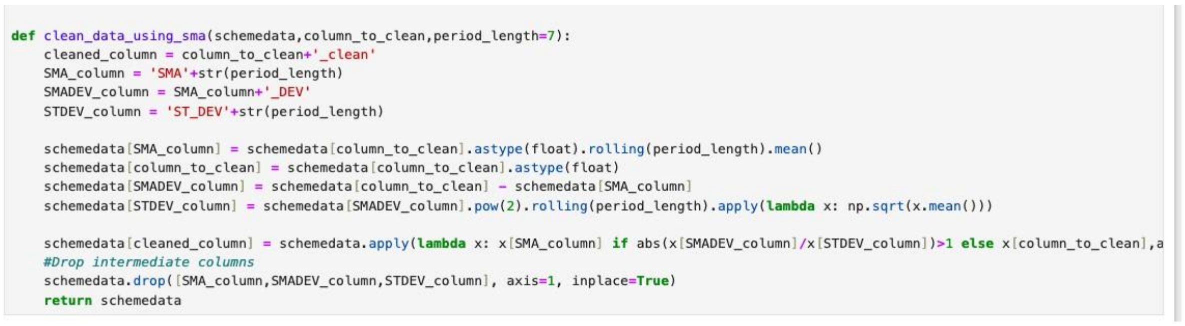
1. Identifying and addressing any inconsistencies, missing values, or anomalies in the dataset to ensure the data is accurate and reliable for further analysis.
2. Summarizing the main features of the dataset through measures such as mean, median, mode, standard deviation, and range. This helps in gaining a preliminary understanding of the data distribution and central tendencies.
3. Creating visual representations of the data, such as histograms, box plots, scatter plots, and line charts, to identify trends, patterns, and relationships between variables. These visual tools make detecting outliers, correlations, and data distributions easier.

4. Examining the relationships between different variables to understand how they influence each other. This can involve calculating correlation coefficients and creating correlation matrices to identify strong and weak relationships.
5. Formulating hypotheses based on the observed data patterns and trends. These hypotheses can guide further analysis and testing in subsequent stages of the project.
6. Applying transformations, such as normalization or scaling, to prepare the data for advanced analysis and modeling. This step ensures that the data is in a suitable format for the next phases of the analysis process.
7. Principal Component Analysis: It can help reduce the number of variables required to represent the data while retaining as much information as possible, thereby compressing large datasets.

8. Code Snippets:

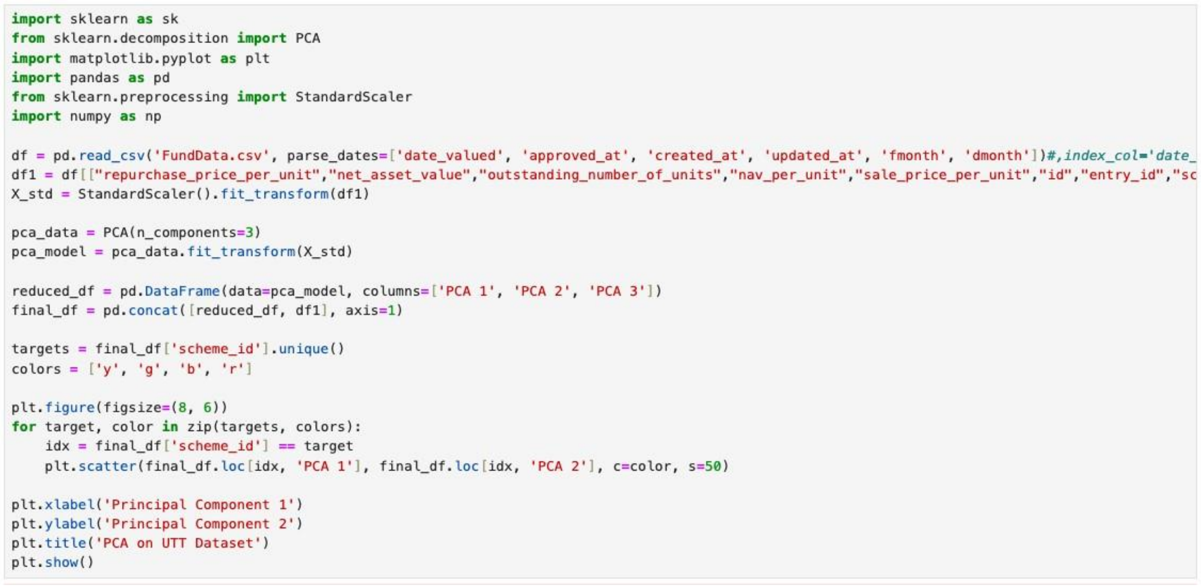


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

#Loading data in the notebook
df = pd.read_csv('MyData_fund.csv')

#EDA/Preprocessing
print(df)

print(df.head())

print(df.describe())

print(df.info())

df.shape

print(df.columns.tolist())

print(df.isnull().sum())

print(df.nunique())

print(df.duplicated().sum())

print(df.tail())

#Statistical Analysis
mean_values = df.mean()
print("Mean:\n", mean_values)

median_values = df.median()
print("Median:\n", median_values)
```

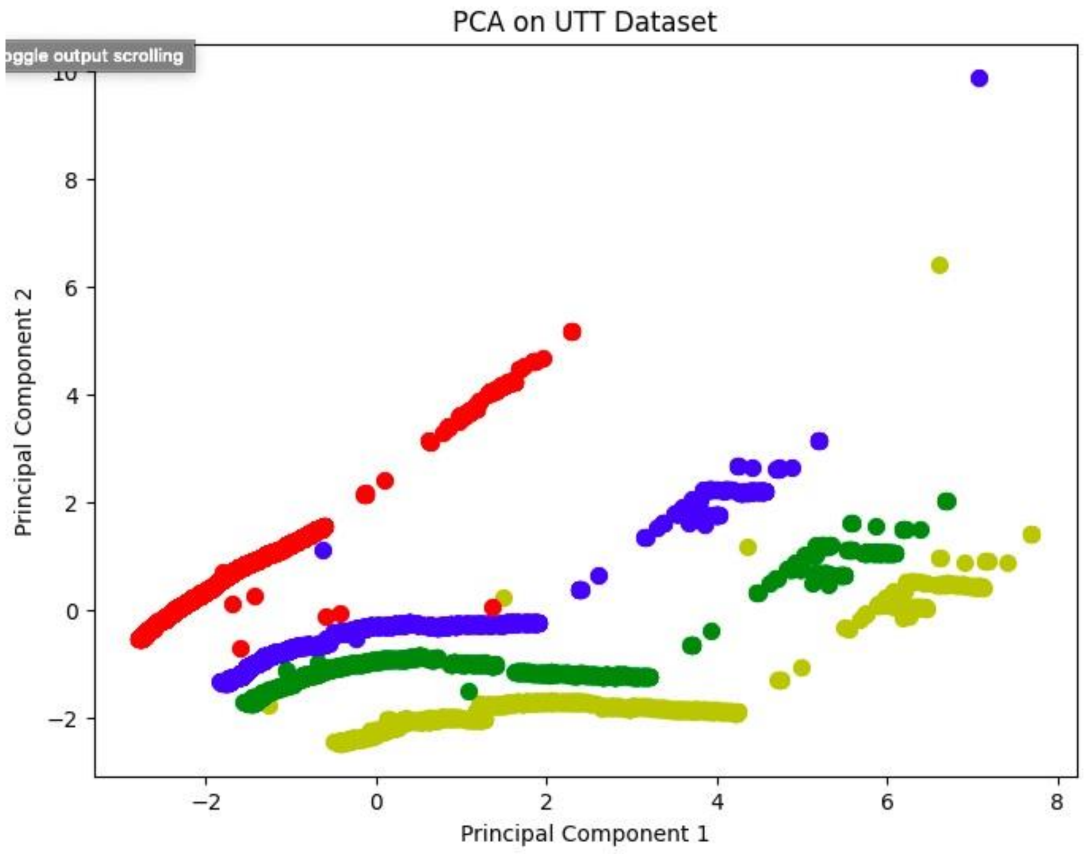


```python
def clean_data_using_sma(schemedata,column_to_clean,period_length=7):
    cleaned_column = column_to_clean+'_clean'
    SMA_column = 'SMA'+str(period_length)
    SMADEV_column = SMA_column+'_DEV'
    STDEV_column = 'ST_DEV'+str(period_length)

    schemedata[SMA_column] = schemedata[column_to_clean].astype(float).rolling(period_length).mean()
    schemedata[column_to_clean] = schemedata[column_to_clean].astype(float)
    schemedata[SMADEV_column] = schemedata[column_to_clean] - schemedata[SMA_column]
    schemedata[STDEV_column] = schemedata[SMADEV_column].pow(2).rolling(period_length).apply(lambda x: np.sqrt(x.mean()))

    schemedata[cleaned_column] = schemedata.apply(lambda x: x[SMA_column] if abs(x[SMADEV_column]/x[STDEV_column])>1 else x[column_to_clean],a
    #Drop intermediate columns
    schemedata.drop([SMA_column,SMADEV_column,STDEV_column], axis=1, inplace=True)
    return schemedata
```

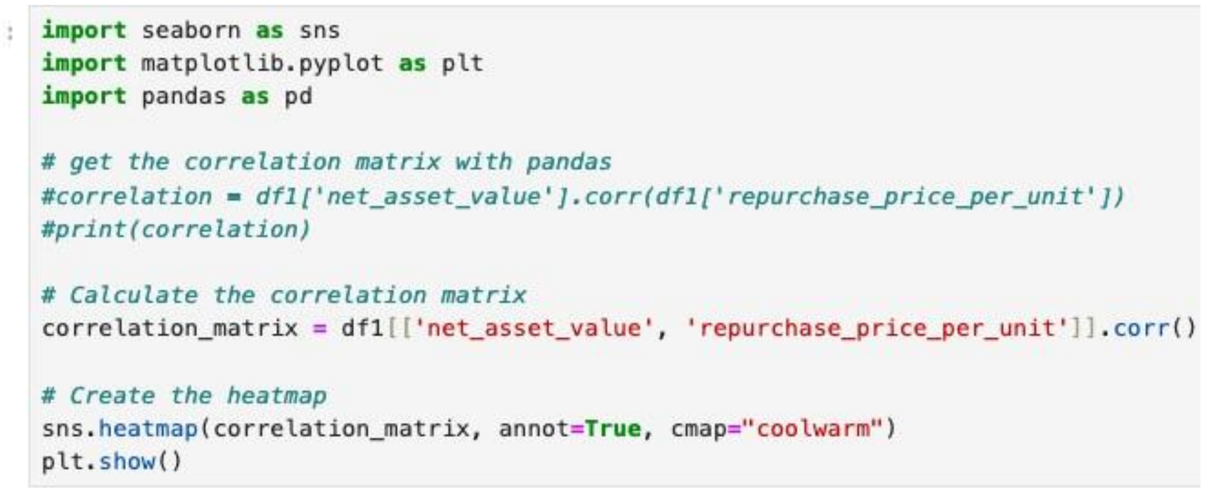
Code Snippet 3: Exploratory Data Analysis on the Dataset



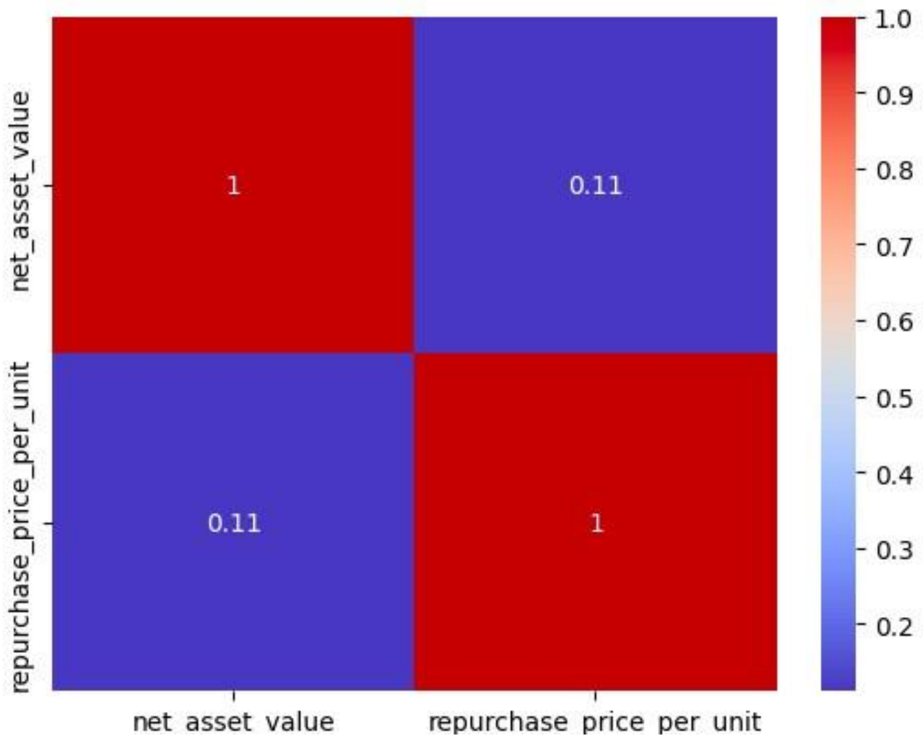
```python
import sklearn as sk
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import StandardScaler
import numpy as np

df = pd.read_csv('FundData.csv', parse_dates=['date_valued', 'approved_at', 'created_at', 'updated_at', 'fmonth', 'dmonth'])#,index_col='date_
df1 = df[["repurchase_price_per_unit","net_asset_value","outstanding_number_of_units","nav_per_unit","sale_price_per_unit","id","entry_id","sc
X_std = StandardScaler().fit_transform(df1)

pca_data = PCA(n_components=3)
pca_model = pca_data.fit_transform(X_std)

reduced_df = pd.DataFrame(data=pca_model, columns=['PCA 1', 'PCA 2', 'PCA 3'])
final_df = pd.concat([reduced_df, df1], axis=1)

targets = final_df['scheme_id'].unique()
colors = ['y', 'g', 'b', 'r']

plt.figure(figsize=(8, 6))
for target, color in zip(targets, colors):
    idx = final_df['scheme_id'] == target
    plt.scatter(final_df.loc[idx, 'PCA 1'], final_df.loc[idx, 'PCA 2'], c=color, s=50)

plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA on UTT Dataset')
plt.show()
```



Code Snippet 4: Principal Component Analysis on the Dataset



```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# get the correlation matrix with pandas
#correlation = df1['net_asset_value'].corr(df1['repurchase_price_per_unit'])
#print(correlation)

# Calculate the correlation matrix
correlation_matrix = df1[['net_asset_value', 'repurchase_price_per_unit']].corr()

# Create the heatmap
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.show()
```

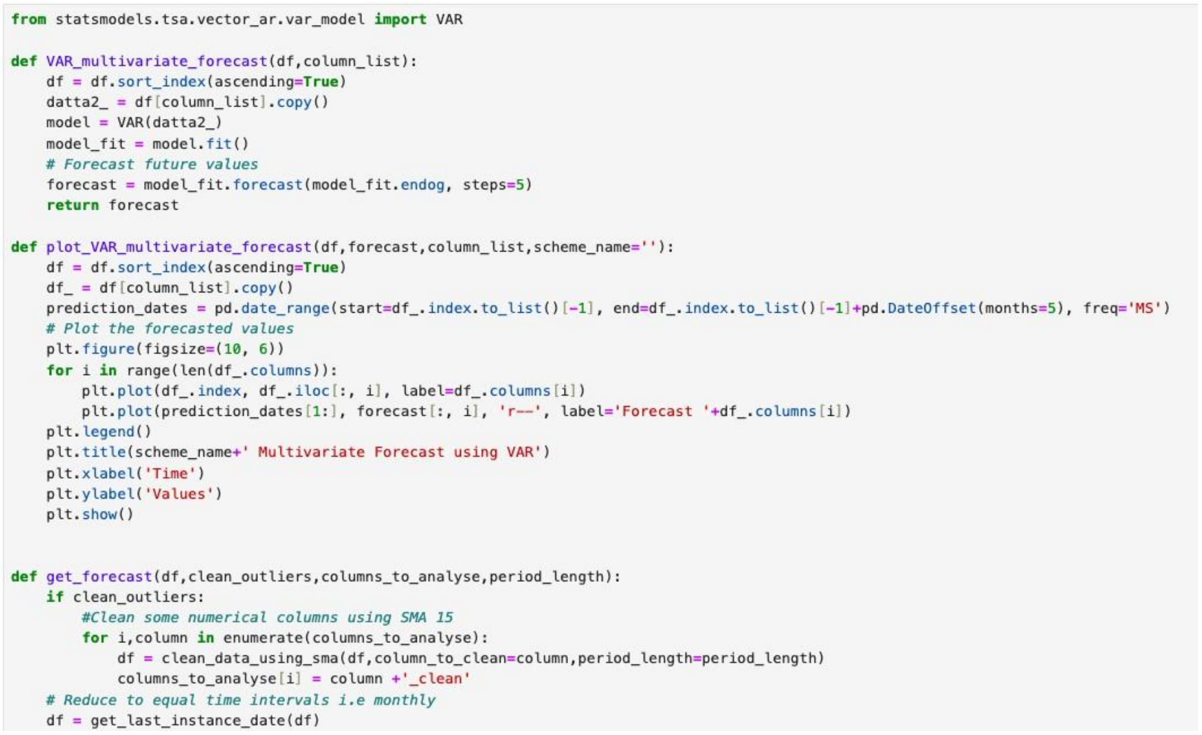


Code Snippet 5: Correlation of prices in a scheme

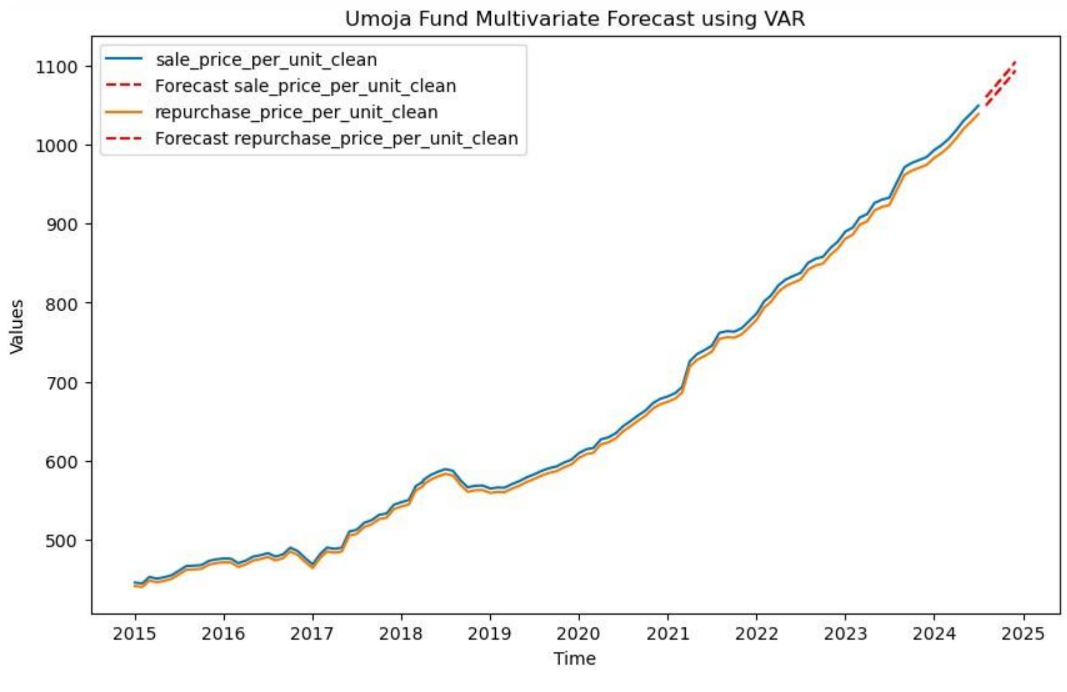
C: PREDICTION PROCESS

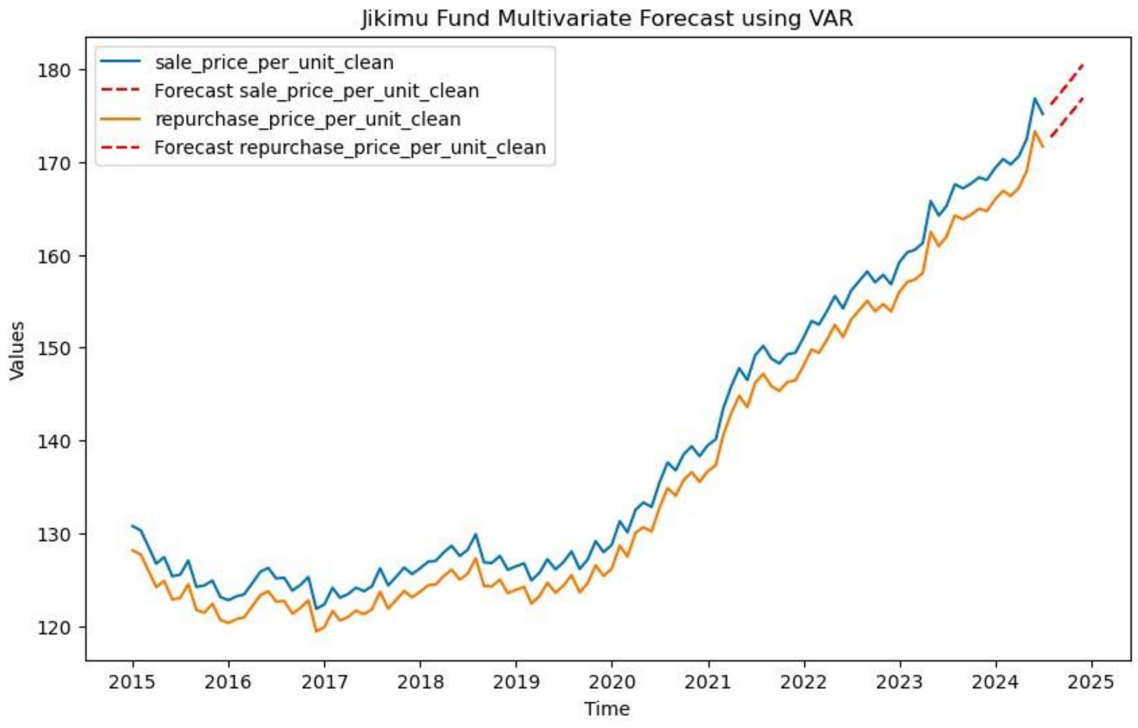
The prediction process involves a series of steps designed to forecast future values based on historical data. This process is critical for making informed investment decisions and strategies. Here's a detailed outline of the prediction process:

a) Identifying the most relevant features (variables) that influence the target variable (e.g., future price) is crucial.
b) Depending on the nature of the data and the prediction goals, appropriate machine learning or statistical models are chosen. Common models for time series and financial data include ARIMA, Vector Autoregressive (VAR), and various machine learning algorithms like Linear Regression, Random Forest, and Gradient Boosting.
c) The selected models are trained using the training dataset. This involves feeding the historical data into the model and allowing it to learn the patterns and relationships between the features and the target variable.
d) Once the model is trained, its performance is evaluated using the testing dataset. Metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R^2) are used to assess the accuracy and reliability of the predictions.
e) The tuned model is further validated using cross-validation techniques to ensure its robustness and reliability across different subsets of the data.
f) Once validated, the model is deployed to make predictions on new data. These predictions provide insights into future trends and help investors make informed decisions.



```python
from statsmodels.tsa.vector_ar.var_model import VAR

def VAR_multivariate_forecast(df,column_list):
    df = df.sort_index(ascending=True)
    datta2_ = df[column_list].copy()
    model = VAR(datta2_)
    model_fit = model.fit()
    # Forecast future values
    forecast = model_fit.forecast(model_fit.endog, steps=5)
    return forecast

def plot_VAR_multivariate_forecast(df,forecast,column_list,scheme_name=''):
    df = df.sort_index(ascending=True)
    df_ = df[column_list].copy()
    prediction_dates = pd.date_range(start=df_.index.to_list()[-1], end=df_.index.to_list()[-1]+pd.DateOffset(months=5), freq='MS')
    # Plot the forecasted values
    plt.figure(figsize=(10, 6))
    for i in range(len(df_.columns)):
        plt.plot(df_.index, df_.iloc[:, i], label=df_.columns[i])
        plt.plot(prediction_dates[1:], forecast[:, i], 'r--', label='Forecast '+df_.columns[i])
    plt.legend()
    plt.title(scheme_name+' Multivariate Forecast using VAR')
    plt.xlabel('Time')
    plt.ylabel('Values')
    plt.show()


def get_forecast(df,clean_outliers,columns_to_analyse,period_length):
    if clean_outliers:
        #Clean some numerical columns using SMA 15
        for i,column in enumerate(columns_to_analyse):
            df = clean_data_using_sma(df,column_to_clean=column,period_length=period_length)
            columns_to_analyse[i] = column +'_clean'
    # Reduce to equal time intervals i.e monthly
    df = get_last_instance_date(df)
```





Code Snippets 6: Prediction of future prices using the VAR model on different schemes of the data

4.1. INTERFACES DESIGNED

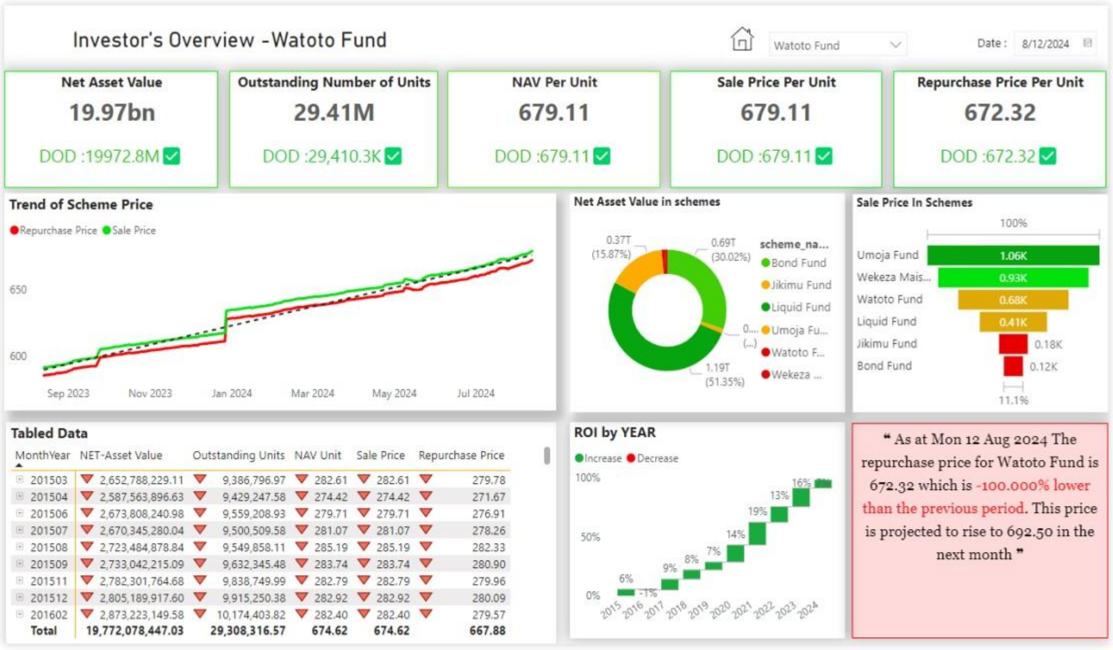
The Power BI dashboard served as the user-friendly interface designed to display the results of the data scraping process from the TAMIS website. It acted as a central hub where all the extracted, processed and analyzed financial data was visually represented, making it easy for users to interact with and understand the information.

The dashboard includes;

1. **Overview Page:**
   - The dashboard began with an overview page summarizing key financial metrics such as total fund performance, average returns, and overall growth trends. This page provided a high-level snapshot, allowing users to quickly grasp the most important data points.
2. **Interactive Visualizations:**
   - The dashboard featured a range of interactive visualizations, including line graphs showing historical performance trends, bar charts comparing different funds, and pie charts illustrating the allocation of assets. Users could hover over these visualizations to see more detailed data or click on specific elements to filter the results.
3. **Filter and Drill-Down Options:**
   - To enable deeper exploration, the dashboard included filter options such as date ranges, fund categories, and performance indicators. This allowed users to drill down into specific datasets, such as examining a particular fund's performance over a selected time frame or comparing different categories side by side.
4. **Data Refresh and Updates:**
   - The Power BI dashboard was connected directly to the backend data source, which meant it could refresh data automatically as new information was scraped from the website. This ensured that the dashboard always displayed the most current and accurate data.
5. **Custom Reports and Insights:**
   - Users could create their custom reports based on the data available in the dashboard, allowing them to focus on the metrics most relevant to their specific needs. Key insights

and trends identified through advanced analysis were highlighted in separate sections to guide decision-making.

6. **User-Friendly Design:**
   - The dashboard was designed with a clear, intuitive layout that prioritized ease of use. Visual elements were color-coded and labeled for clarity, and the overall design followed best practices for data visualization to enhance comprehension.



Net Asset Value	Outstanding Number of Units	NAV Per Unit	Sale Price Per Unit	Repurchase Price Per Unit
19.97bn	29.41M	679.11	679.11	672.32
DOD :19972.8M ✓	DOD :29,410.3K ✓	DOD :679.11 ✓	DOD :679.11 ✓	DOD :672.32 ✓

Tabled Data

MonthYear	NET-Asset Value	Outstanding Units	NAV Unit	Sale Price	Repurchase Price
201503	2,652,788,229.11	9,386,796.97	282.61	282.61	279.78
201504	2,587,563,896.63	9,429,247.58	274.42	274.42	271.67
201506	2,673,808,240.98	9,559,208.93	279.71	279.71	276.91
201507	2,670,345,280.04	9,500,509.58	281.07	281.07	278.26
201508	2,723,484,878.84	9,549,858.11	285.19	285.19	282.33
201509	2,733,042,215.09	9,632,345.48	283.74	283.74	280.90
201511	2,782,301,764.68	9,838,749.99	282.79	282.79	279.96
201512	2,805,189,917.60	9,915,250.38	282.92	282.92	280.09
201602	2,873,223,149.58	10,174,403.82	282.40	282.40	279.57
Total	19,772,078,447.03	29,308,316.57	674.62	674.62	667.88

" As at Mon 12 Aug 2024 The repurchase price for Watoto Fund is 672.32 which is -100.000% lower than the previous period. This price is projected to rise to 692.50 in the next month "

Picture 1. Overview Page of the Power BI Dashboard displaying different visuals of the data.

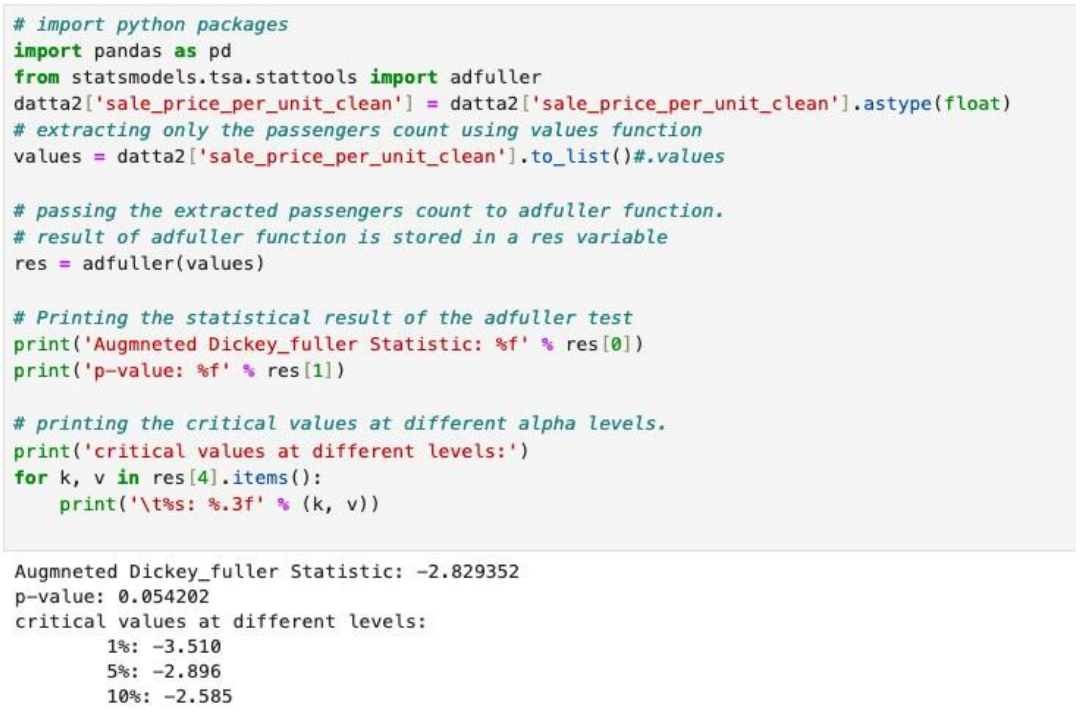
CHALLENGES FACED AND HOW THEY WERE TACKLED

1. CHALLENGES

In the context of our asset management and investment data analysis project, several challenges can arise during the forecasting process. Understanding and addressing these challenges is crucial for improving the accuracy and reliability of our forecasts.

Here are some of the key challenges:

1. Handling Stationarity and Trends:
   - Identifying and accounting for seasonal patterns and long-term trends in the data. Used the ADF (Augmented-Dickey Fuller) Test to check for stationarity and conversion of the data to stationary.
   - The data agreed with the Null Hypothesis, that the data was non-stationary but also converted to stationary data; the statistic is negative, indicating evidence in favor of stationarity.



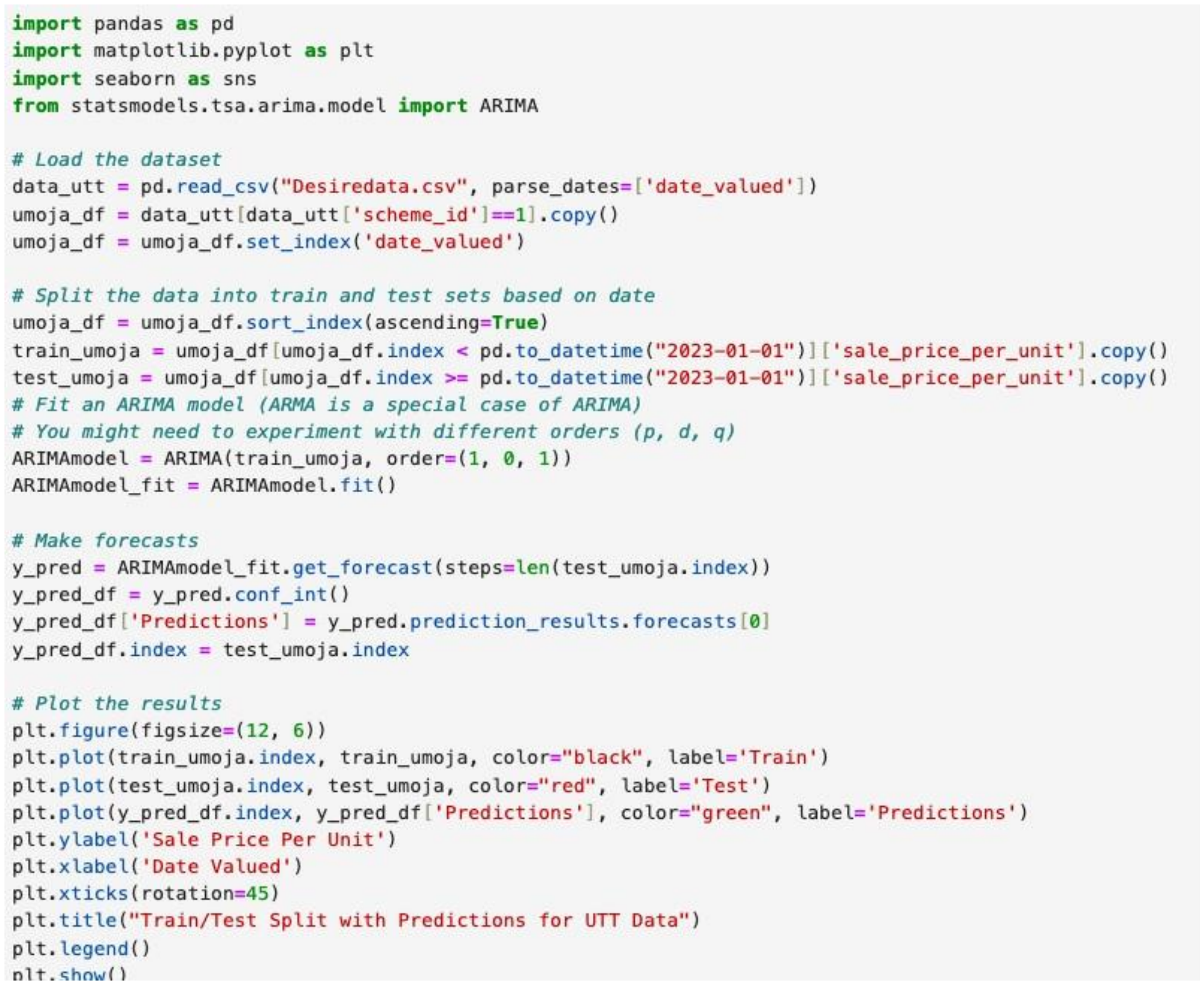
```python
# import python packages
import pandas as pd
from statsmodels.tsa.stattools import adfuller
datta2['sale_price_per_unit_clean'] = datta2['sale_price_per_unit_clean'].astype(float)
# extracting only the passengers count using values function
values = datta2['sale_price_per_unit_clean'].to_list()#.values

# passing the extracted passengers count to adfuller function.
# result of adfuller function is stored in a res variable
res = adfuller(values)

# Printing the statistical result of the adfuller test
print('Augmneted Dickey_fuller Statistic: %f' % res[0])
print('p-value: %f' % res[1])

# printing the critical values at different alpha levels.
print('critical values at different levels:')
for k, v in res[4].items():
    print('\t%s: %.3f' % (k, v))
```

```
Augmneted Dickey_fuller Statistic: -2.829352
p-value: 0.054202
critical values at different levels:
        1%: -3.510
        5%: -2.896
        10%: -2.585
```

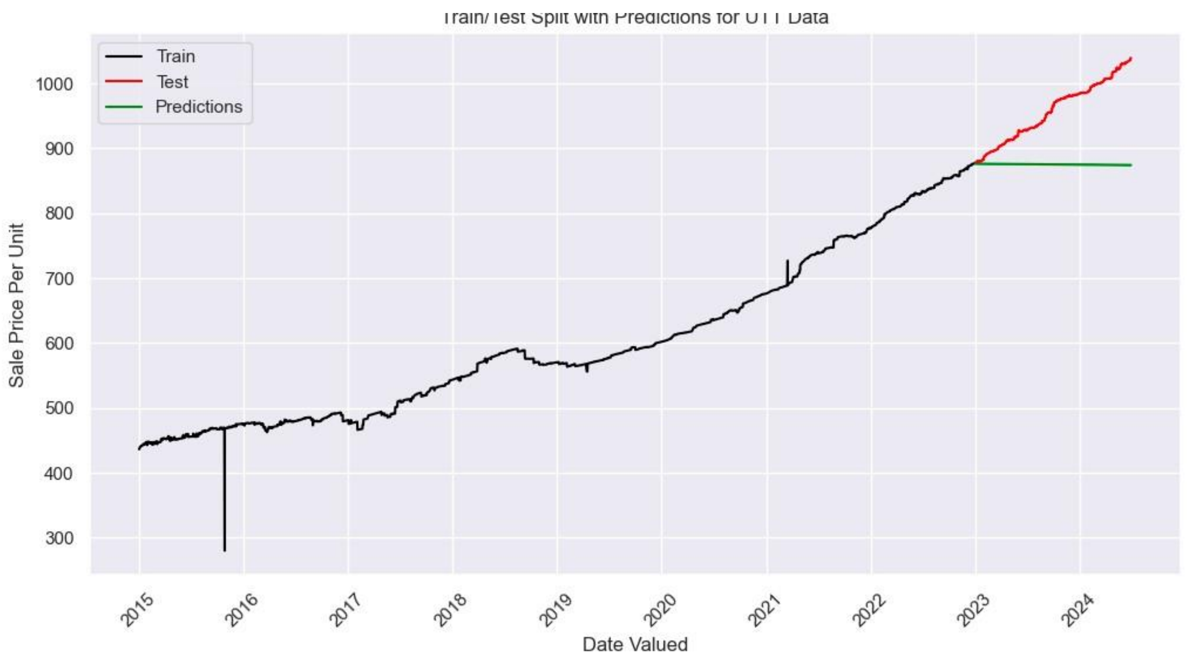
2. Selecting the Right Forecasting Model:

- Choosing the appropriate forecasting model that accurately captures the underlying trends and seasonality of the data.
- Experimenting with different models, such as ARIMA, exponential smoothing, and machine learning algorithms, and validating their performance using historical data.

The ARIMA Model prediction:



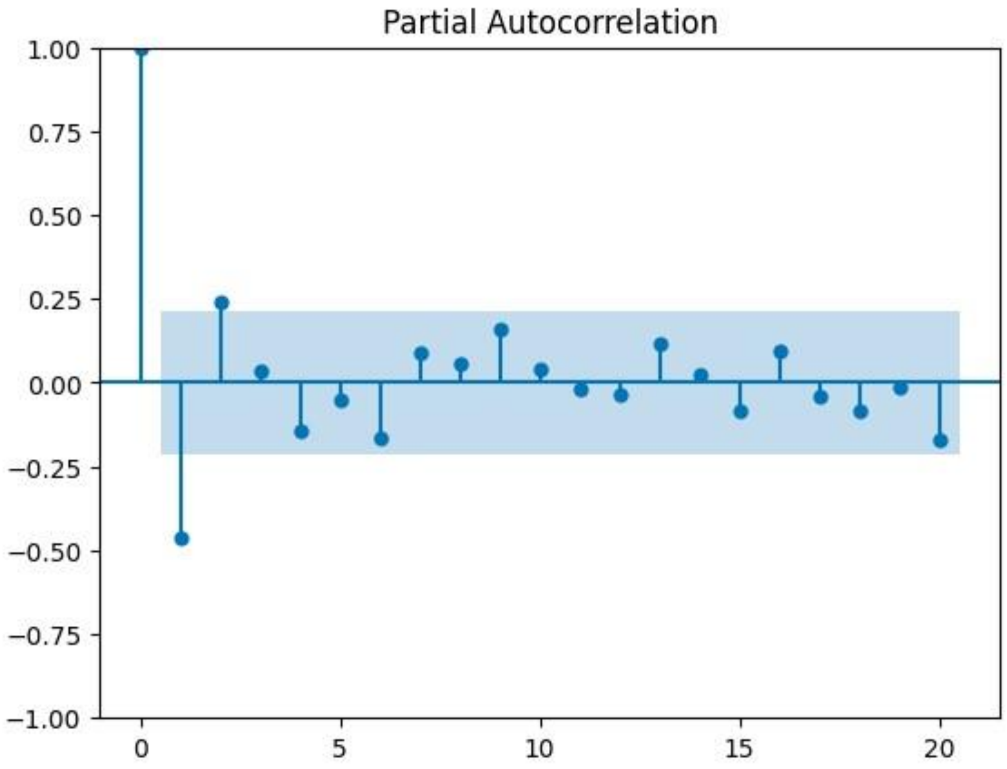
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.arima.model import ARIMA

# Load the dataset
data_utt = pd.read_csv("Desiredata.csv", parse_dates=['date_valued'])
umoja_df = data_utt[data_utt['scheme_id']==1].copy()
umoja_df = umoja_df.set_index('date_valued')

# Split the data into train and test sets based on date
umoja_df = umoja_df.sort_index(ascending=True)
train_umoja = umoja_df[umoja_df.index < pd.to_datetime("2023-01-01")]['sale_price_per_unit'].copy()
test_umoja = umoja_df[umoja_df.index >= pd.to_datetime("2023-01-01")]['sale_price_per_unit'].copy()
# Fit an ARIMA model (ARMA is a special case of ARIMA)
# You might need to experiment with different orders (p, d, q)
ARIMAmodel = ARIMA(train_umoja, order=(1, 0, 1))
ARIMAmodel_fit = ARIMAmodel.fit()

# Make forecasts
y_pred = ARIMAmodel_fit.get_forecast(steps=len(test_umoja.index))
y_pred_df = y_pred.conf_int()
y_pred_df['Predictions'] = y_pred.prediction_results.forecasts[0]
y_pred_df.index = test_umoja.index

# Plot the results
plt.figure(figsize=(12, 6))
plt.plot(train_umoja.index, train_umoja, color="black", label='Train')
plt.plot(test_umoja.index, test_umoja, color="red", label='Test')
plt.plot(y_pred_df.index, y_pred_df['Predictions'], color="green", label='Predictions')
plt.ylabel('Sale Price Per Unit')
plt.xlabel('Date Valued')
plt.xticks(rotation=45)
plt.title("Train/Test Split with Predictions for UTT Data")
plt.legend()
plt.show()
```



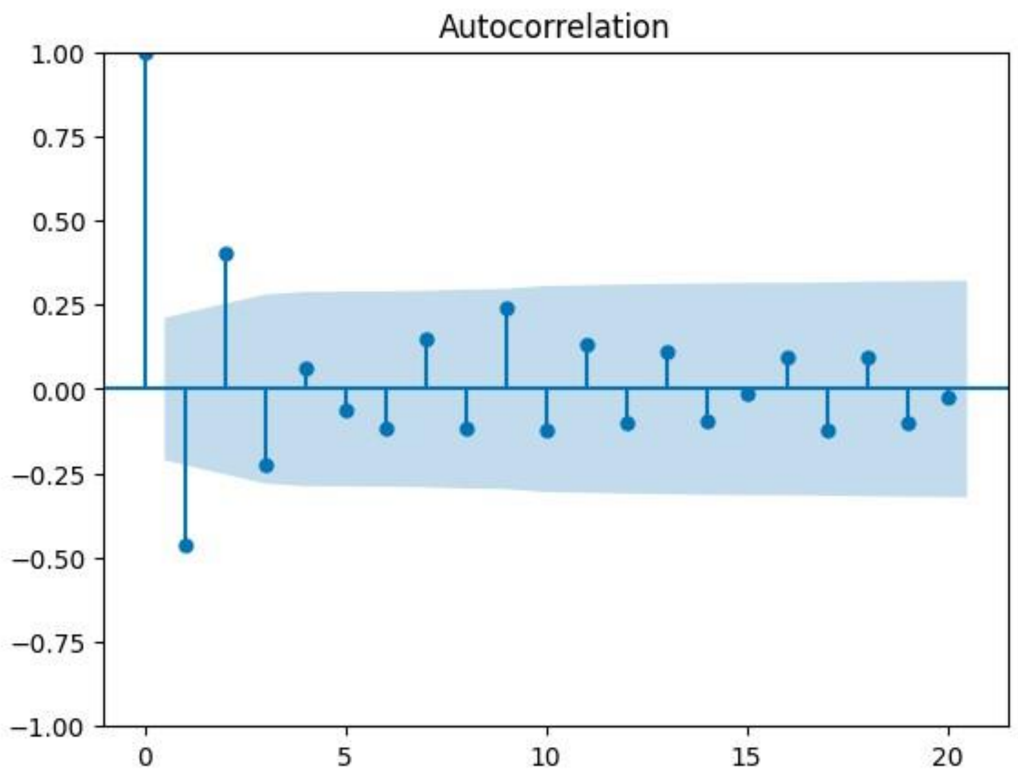
The non-stationarity brought about the inconsistency of the predictions of the data as shown in the above plot.

The measure of P, D, and Q order values for the conversion of data to stationarity.

```python
from statsmodels.graphics.tsaplots import plot_pacf
plot_pacf(datta2['sale_price_per_unit_clean'].diff().dropna());
```



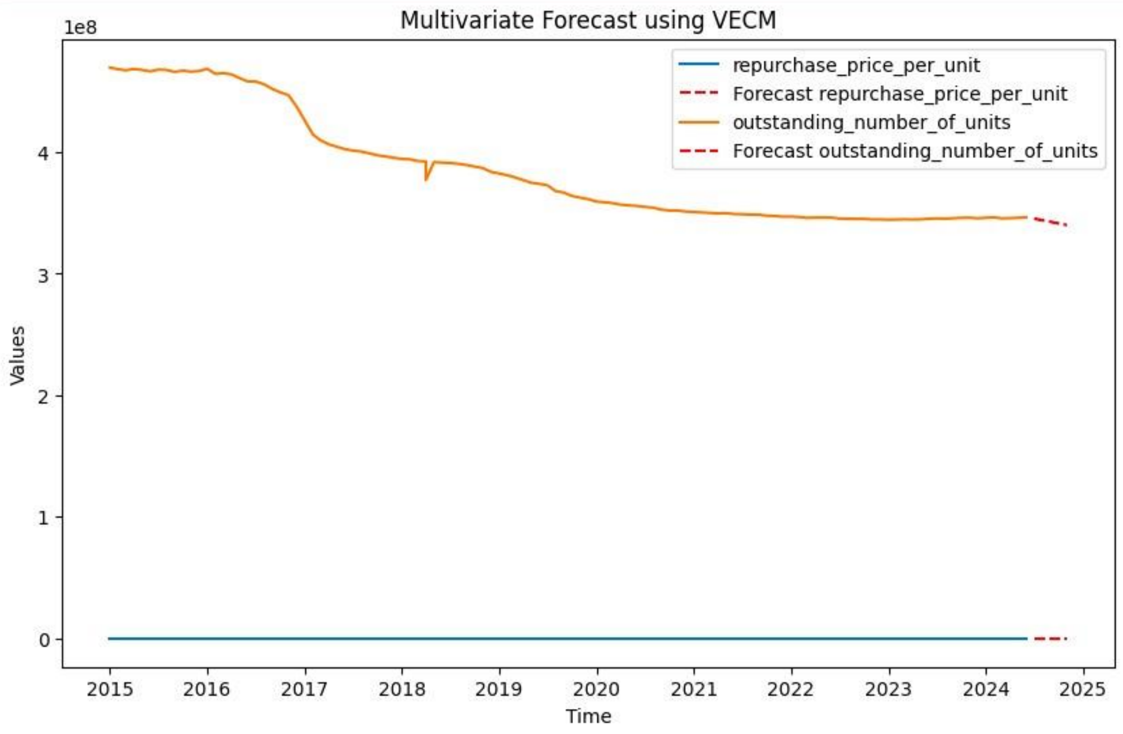
```python
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(datta2['sale_price_per_unit_clean'].diff().dropna());
```



Vector Error Correction Model (VECM) for Prediction

The Vector Error Correction Model (VECM) is a powerful tool for modeling and forecasting multivariate time series data, particularly when the data series are cointegrated. Cointegration implies that the data series share a long-term equilibrium relationship despite being non-stationary individually.

```python
# Importing necessary libraries
import pandas as pd
from statsmodels.tsa.vector_ar.vecm import VECM
import matplotlib.pyplot as plt
# Load multivariate time series data
#data = pd.read_csv('VECM-multivariate_data.csv')

columnlist = ["repurchase_price_per_unit","outstanding_number_of_units"]
#columnlist = ["net_asset_value","outstanding_number_of_units"]

data1 = get_last_instance_date(schemedata)
data1 = data1[columnlist].copy()
data1 = data1.sort_index(ascending=True)

# Fit the VECM model
model = VECM(data1)
model_fit = model.fit()
# Forecast future values
forecast = model_fit.predict(steps=5)
prediction_dates = pd.date_range(start=data1.index.to_list()[-1], end=data1.index.to_list()[-1]+pd.DateOffset(months=5), freq='MS')
# Plot the forecasted values
plt.figure(figsize=(10, 6))
for i in range(len(data1.columns)):
    plt.plot(data1.index, data1.iloc[:, i], label=data1.columns[i])
    plt.plot(prediction_dates[1:], forecast[:, i], 'r--', label='Forecast '+data1.columns[i])
    #plt.plot(range(len(data), len(data) + 5), forecast[:, i], 'r--', label='Forecast '+data.columns[i])
plt.legend()
plt.title('Multivariate Forecast using VECM')
plt.xlabel('Time')
plt.ylabel('Values')
plt.show()
```



3. Overfitting and Underfitting

Balancing the model to avoid overfitting (where the model learns the noise in the data) and underfitting (where the model is too simple to capture the underlying trend) is another significant challenge. Overfitting can lead to excellent performance on training data but poor generalization to new data while underfitting results in a model that fails to capture important patterns in the data.

4.

THE FINDINGS AND DISCUSSIONS

Summary of Findings

The exploratory data analysis (EDA) and forecasting process revealed several key findings:

1. **Strong positive correlations**: The features are strongly positively correlated, indicating that they are related to each other.
2. **Increasing trend**: The time series plot reveals an increasing trend in the data, indicating that the values are increasing over time.
3. **Skewed distribution**: The data is skewed to the right, indicating that there are outliers present.
4. **Inaccurate forecasting**: The Random Forest Regressor model was not able to capture the trend and seasonality of the data, and the predicted values deviated from the actual values.

Discussion

The strong positive correlations between the features suggest that they are interdependent and can be used to predict each other. This is consistent with the idea that the sale price and repurchase price are related to each other in the context of the website.

The increasing trend in the data suggests that the website's performance is improving over time. This could be due to various factors such as changes in marketing strategies, improvements in user experience, or demand increases.

The skewed distribution of the data suggests that there are outliers present, which could be due to various factors such as errors in data collection or unusual events. The presence of outliers can affect the accuracy of the forecasting model, and therefore, it is essential to handle them appropriately.

The inaccurate forecasting results suggest that the Random Forest Regressor model is not a good fit for the data. This model can't be used to predict future values of the repurchase price per unit, which can inform business decisions and improve the website's performance.

Implications

The findings of this study have several implications for the website:

1. **Price optimization**: The strong positive correlations between the features suggest that optimizing one price can positively impact the others. Therefore, the website can use this information to optimize its pricing strategy.
2. **Trend analysis**: The increasing trend in the data suggests that the website's performance is improving over time. Therefore, the website can use this information to identify areas of improvement and make data-driven decisions.
3. **Outlier detection**: The presence of outliers in the data suggests that the website needs to implement measures to detect and handle outliers effectively.

Limitations

This study has several limitations:

1. **Data quality**: The quality of the data used in this study may not be perfect, which can affect the accuracy of the results.
2. **Model selection**: The Random Forest Regressor model was selected based on its performance, but the Vector Autoregression model was suitable for the data.
3. **Feature engineering**: The features used in this study may not be the most relevant or effective features for prediction purposes.

Future Research Directions

This study suggests several directions for future research:

1. **Feature engineering**: Investigating the effect of different features on the desired predictive feature before the forecasting process.
2. **Model selection**: Comparing the performance of different models on the data.
3. **Data quality improvement**: Improving the quality of the data used in this study to increase the accuracy of the results.

Bibliography

1. Microsoft Corporation. "Power BI Documentation." Power BI Overview, Microsoft Learn. Accessed on 23 Jul 2024.
2. McKinney, Wes. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media, 2017.
3. TAMIS Fund Performance. "Tanzania Association of Mutual Funds." Fund Performance Overview, Tanzania Association of Mutual Funds, Accessed on 24 Jun 2024.
4. Hastie, Trevor, Tibshirani, Robert, and Friedman, Jerome. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed., Springer, 2009.
5. Wickham, Hadley, and Grolemund, Garrett. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. O'Reilly Media, 2016.