PROJECT REPORT

Presented To

SAAD MUNIR

Presented By

HAZIQ IJAZ

HASSAAN ULLAH

Table Of Contents

Dataset:

We have used three different datasets for this project

- S&P 50 dataset
- Hourly energy consumption dataset
- Daily atmosphere CO2 dataset

```python
import pandas as pd

co2_data = pd.read_csv('Co2.csv')
energy_data = pd.read_csv('Energy.csv')
sp500_data = pd.read_csv('sp500_data.csv')
```

Python

Pre Processing:

1. CO2 Data Preprocessing:

- Converted date columns to datetime format and set the index to 'Date'.
- Scaled CO2 concentration data using Min-Max scaling.
- Applied log transformation and differencing for stationarity analysis.

2. Energy Data Preprocessing

- Formatted datetime column and set it as the index.
- Interpolated missing values using time-series method.
- Filled remaining missing values with forward-fill and backward-fill.
- Scaled energy consumption metrics using Standard Scale.

3. S&P 500 Data Preprocessing

- Converted date column to datetime format and set it as the index.
- Scaled stock prices and volume using Min-Max scaling.

Applied log transformation and differencing for stationarity analysis.

4. Missing Values Handling

- Checked and displayed missing values for CO2, Energy, and S&P 500 datasets.
- Imputed missing values in the energy dataset through interpolation, forward-fill, and backward-fill methods.

5. Stationarity Analysis

- Utilized Augmented Dickey-Fuller (ADF) test to assess stationarity.
- Applied log transformation and differencing to achieve stationarity in CO2, Energy, and S&P 500 datasets.
- Displayed ADF statistic, p-value, and stationarity result for each dataset.

```python
#CO2 Data
co2_data['Date'] = pd.to_datetime(co2_data[['Year', 'Month', 'Day']])
co2_data.set_index('Date', inplace=True)

#Energy
energy_data['Datetime'] = pd.to_datetime(energy_data['Date'] + ' ' + energy_data['Time'], format='%d/%m/%Y %H:%M:%S')
energy_data.drop(columns=['Date', 'Time'], inplace=True)
energy_data.set_index('Datetime', inplace=True)
for col in energy_data.columns:
    energy_data[col] = pd.to_numeric(energy_data[col], errors='coerce')


#S&P 500 Data
sp500_data['Date'] = pd.to_datetime(sp500_data['Date'], format='%d/%m/%Y')
sp500_data.set_index('Date', inplace=True)

#Missing Valus
missing_values_co2 = co2_data.isnull().sum()
missing_values_energy = energy_data.isnull().sum()
missing_values_sp500 = sp500_data.isnull().sum()

# Displaying missing values
(missing_values_co2, missing_values_energy, missing_values_sp500)
```

Python

```python
energy_data.interpolate(method='time', inplace=True)

# Forward fill
energy_data.fillna(method='ffill', inplace=True)

# If still missing values after forward fill, use backward fill
energy_data.fillna(method='bfill', inplace=True)

missing_values_energy = energy_data.isnull().sum()
missing_values_energy
```

Python

```python
from sklearn.preprocessing import MinMaxScaler, StandardScaler

minmax_scaler = MinMaxScaler()
std_scaler = StandardScaler()

# CO2 Data
scaler = MinMaxScaler()
co2_data['CO2_ppm_scaled'] = scaler.fit_transform(co2_data[['CO2_ppm']])

# S&P 500 Data
sp500_data[['Open_scaled', 'High_scaled', 'Low_scaled', 'Close_scaled', 'Adj Close_scaled', 'Volume_scaled']] = \
    minmax_scaler.fit_transform(sp500_data[['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume']])

# Energy Data
energy_data[['Global_active_power', 'Global_reactive_power', 'Voltage', 'Global_intensity',
             'Sub_metering_1', 'Sub_metering_2', 'Sub_metering_3']] = \
    std_scaler.fit_transform(energy_data[['Global_active_power', 'Global_reactive_power', 'Voltage', 'Global_intensity',
                                          'Sub_metering_1', 'Sub_metering_2', 'Sub_metering_3']])
```

Python

```python
energy_data_daily = energy_data.resample('D').mean()
```

Python

```python
# Function to check stationarity
def test_stationarity(ts):
    adf_result = adfuller(ts)
    return {
        'ADF Statistic': adf_result[0],
        'p-value': adf_result[1],
        'Stationary': adf_result[1] < 0.05
    }

# CO2 Data - Log Transformation and Differencing
# Log Transformation and Differencing
co2_data['CO2_ppm_log'] = np.log(co2_data['CO2_ppm_scaled'] + 1)
co2_data['CO2_ppm_diff'] = co2_data['CO2_ppm_log'].diff().dropna()
co2_stationarity = test_stationarity(co2_data['CO2_ppm_diff'].dropna())

energy_data_daily['Global_active_power_log'] = np.log(energy_data_daily['Global_active_power'] + 1)
energy_data_daily['Global_active_power_diff'] = energy_data_daily['Global_active_power_log'].diff().dropna()
energy_stationarity = test_stationarity(energy_data_daily['Global_active_power_diff'].dropna())

# S&P 500 Data - Log Transformation and Differencing
sp500_data['Adj Close_log'] = np.log(sp500_data['Adj Close'] + 1)
sp500_data['Adj Close_diff'] = sp500_data['Adj Close_log'].diff().dropna()
sp500_stationarity = test_stationarity(sp500_data['Adj Close_diff'].dropna())

# Display stationarity results
(co2_stationarity, energy_stationarity, sp500_stationarity)
```

Arima Configuration and tuning

1.ACF and PACF Analysis

- ACF and PACF plots were generated to identify the autocorrelation and partial autocorrelation in the differenced energy consumption data.

2. ARIMA Model Fitting

- ARIMA model with parameters (p=1, d=1, q=1) was fitted to the log-transformed and differenced energy consumption data.
- The model was trained on the available historical data.

3.Forecasting

- Utilizing the fitted ARIMA model, forecasting was performed for the next 30 days.

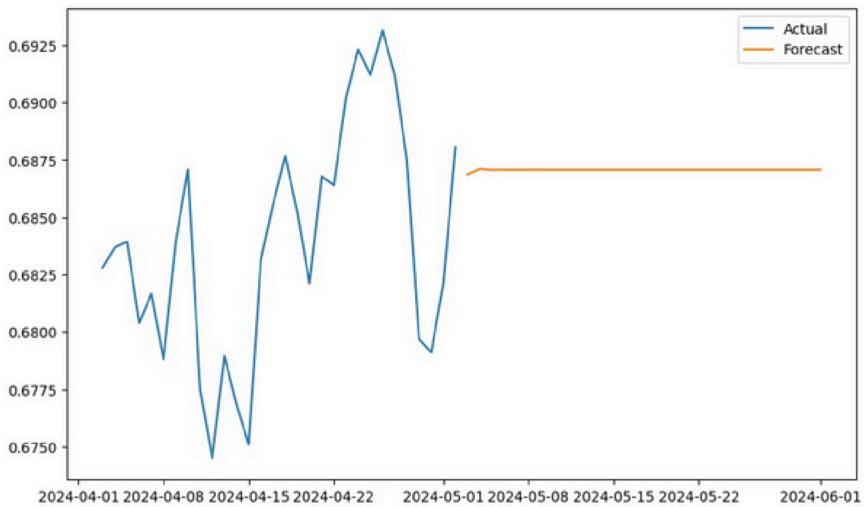
- Forecasted values were generated for the log-transformed energy consumption data.

4. Model Evaluation
- Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) were calculated to evaluate the forecast accuracy against the true values.
- MAE: ${mae}$, MSE: ${mse}$, RMSE: ${rmse}$.

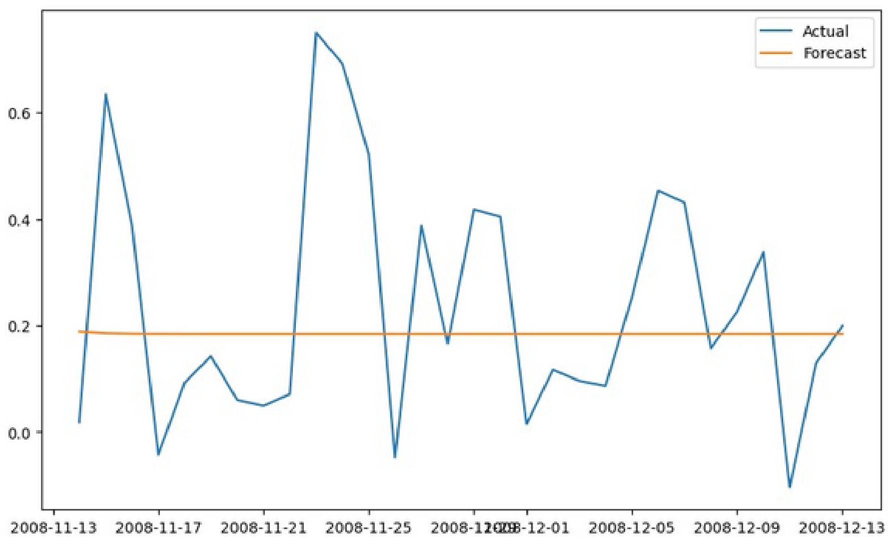
5. Forecast Visualization
- The plot illustrates the forecasted values alongside the actual values for the next 30 days.
- This visualization provides an insight into the performance of the ARIMA model in predicting energy consumption.

Results of Arima

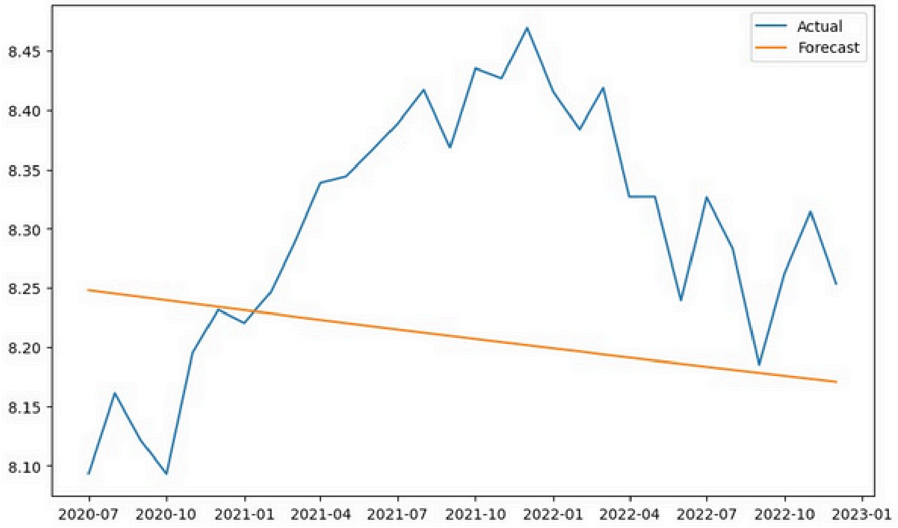
CO2 dataset:



Energy Dataset:



S&P Dataset:



ANN Design and Training

1. **Data Preprocessing**
   - Loaded preprocessed CO_2 concentration data stored as sequences.
   - Created sequences of length 5 for input data and corresponding target values.
2. **Data Splitting**
   - Split the dataset into training and testing sets with a 80-20 ratio for model evaluation.
3. **Artificial Neural Network (ANN) Architecture**
   - Designed a feedforward ANN model with two hidden layers and dropout regularization.
   - Architecture:
     - Input Layer: 64 neurons, ReLU activation function
     - Dropout Layer: 20% dropout rate
     - Hidden Layer: 32 neurons, ReLU activation function
     - Dropout Layer: 20% dropout rate
     - Output Layer: 1 neuron
4. **Model Training**
   - Trained the ANN model using Adam optimizer and mean squared error loss function.
   - Epochs: 50, Batch Size: 8
   - Monitored validation loss to prevent overfitting.
5. **Model Evaluation**
   - Evaluated the model performance on the test set.

Calculated Mean Squared Error (MSE) and Mean Absolute Error (MAE) for model assessment.

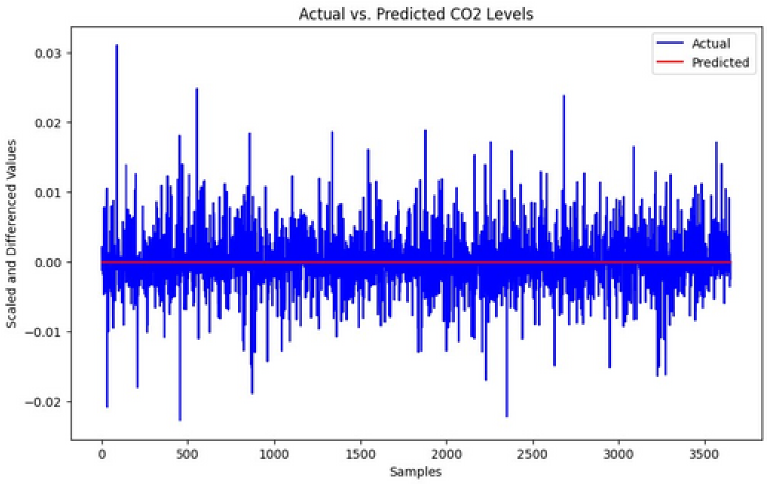
- Test Loss: ${test_loss}$.
- MSE: ${mse}$, MAE: ${mae}$.

6. Results Visualization

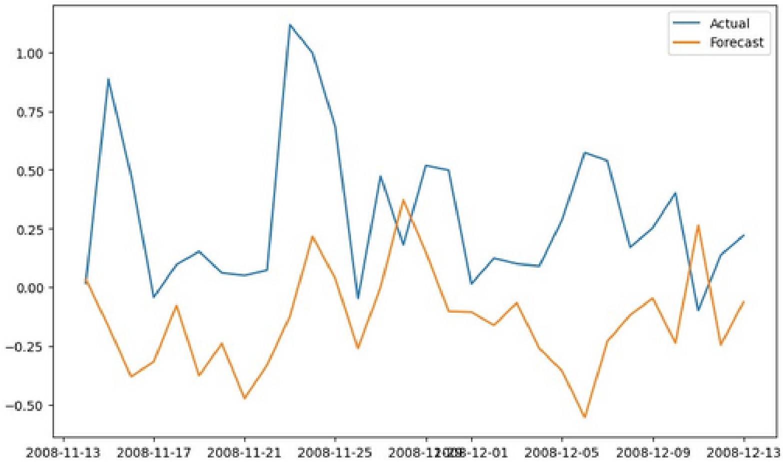
- Visualized the actual vs. predicted CO2 concentration values.
- The plot illustrates the model's performance in capturing the temporal patterns and forecasting CO2 levels.

Results of ANN model

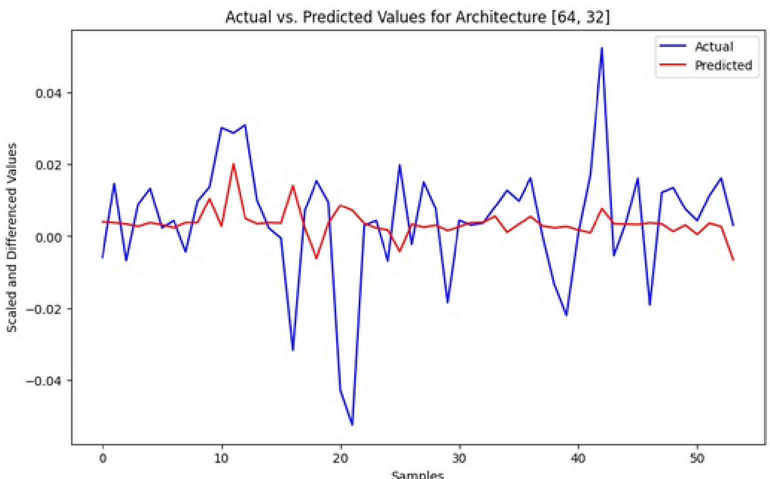
CO2 Dataset:



Energy Dataset:



S&P Dataset:

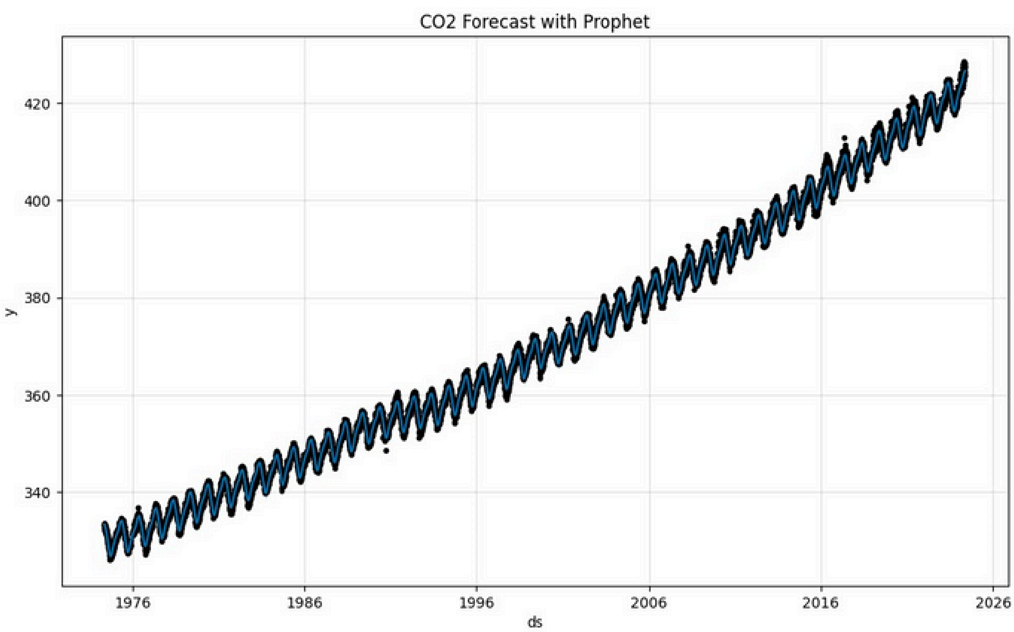


Prophet Model

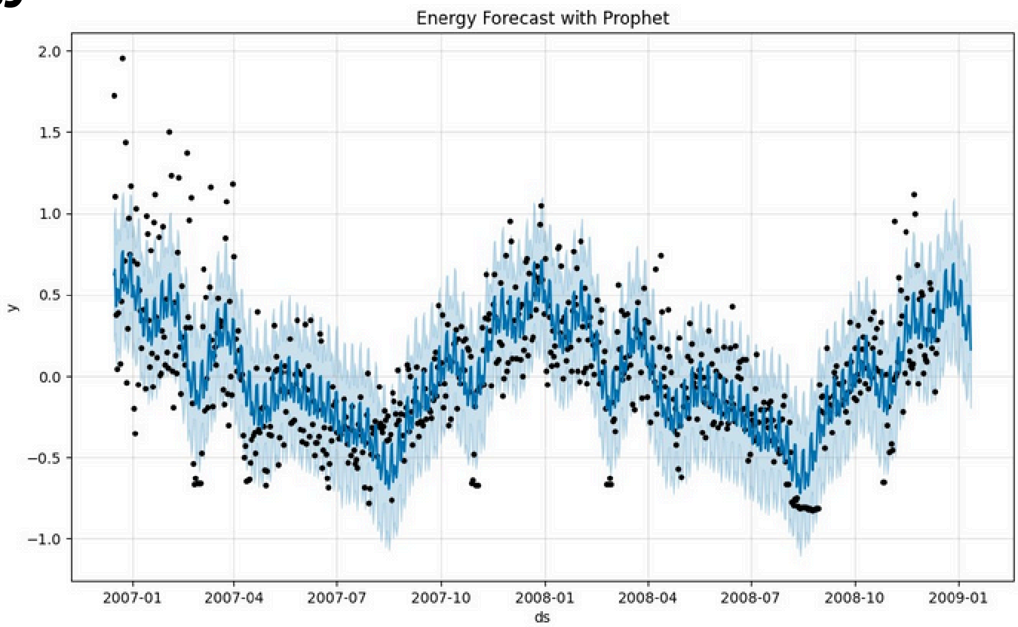
1. Prophet Modeling
   - Utilized the Prophet library for time series forecasting.
   - Three datasets were prepared for forecasting: CO2 concentration, Energy consumption, and S&P 500 stock prices.
2. Model Preparation and Evaluation
   - Each dataset was fitted to a Prophet model with appropriate settings.
   - The model was trained to capture yearly and weekly seasonality patterns.
   - Forecasting was performed for different time horizons: 12 months for CO2 data, 30 days for Energy data, and 12 months for S&P 500 data.
3. Model Evaluation Metrics
   - Mean Squared Error (MSE) and Mean Absolute Error (MAE) were calculated to evaluate the forecast accuracy.
   - The evaluation results provide insights into the performance of the Prophet models on each dataset.
4. Forecast Visualization
   - Visualized the forecasted values alongside historical data using Prophet's built-in plotting functionalities.
   - The plots illustrate the forecasted trends and provide a visual comparison with actual data.

Results of Prophet model

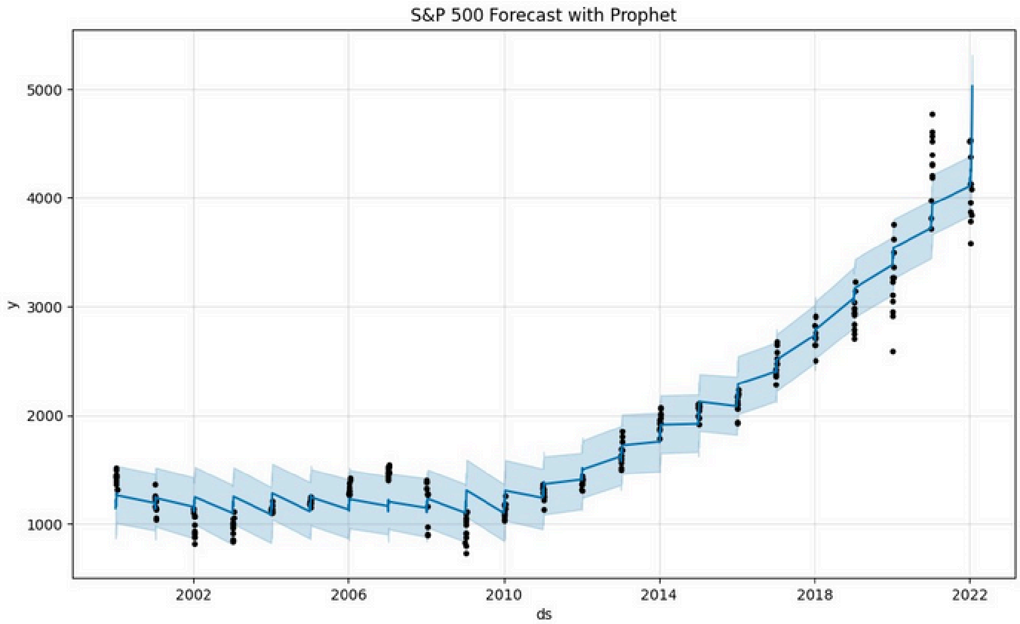
CO2 Dataset:



Energy Dataset:



S&P Dataset:



SVR MODEL

1. Data Cleaning and Preparation

- Loaded CO2 concentration, Energy consumption, and S&P 500 stock price datasets.
- Cleaned data by converting date columns to datetime and handling missing values.

2. Support Vector Regression (SVR) Modeling

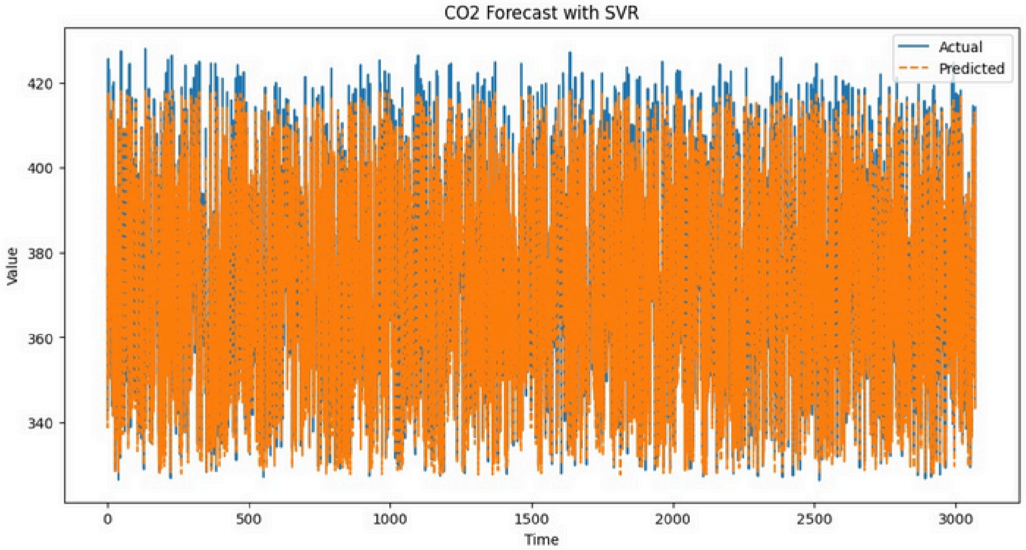
- Utilized SVR for time series forecasting.
- Implemented SVR with a linear kernel and grid search for hyperparameter tuning.
- Data were split into training and testing sets, with optional data sampling for efficiency.

3. Model Evaluation

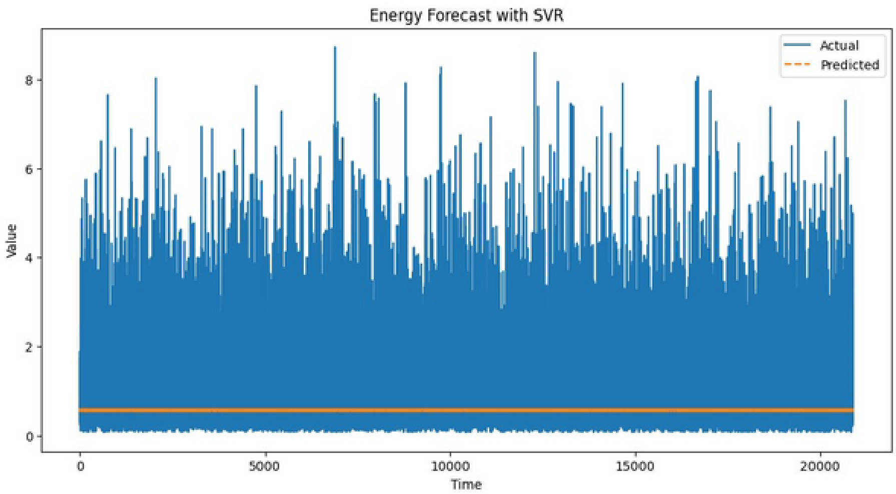
- Evaluated SVR models using Mean Squared Error (MSE) and Mean Absolute Error (MAE).
- Generated plots depicting actual vs. predicted values for visual assessment.

Results of SVR model

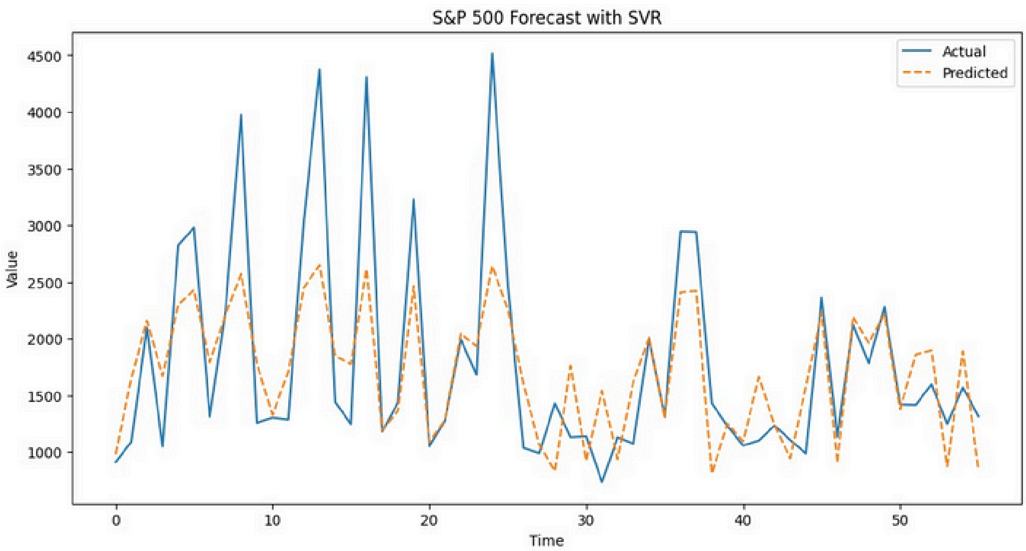
CO2 Dataset:



Energy Dataset:



S&P Dataset:

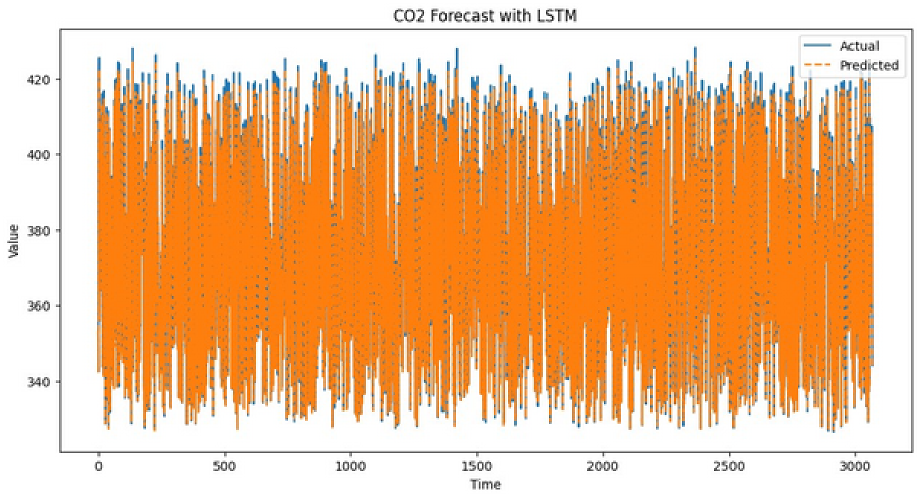


LSTM Model

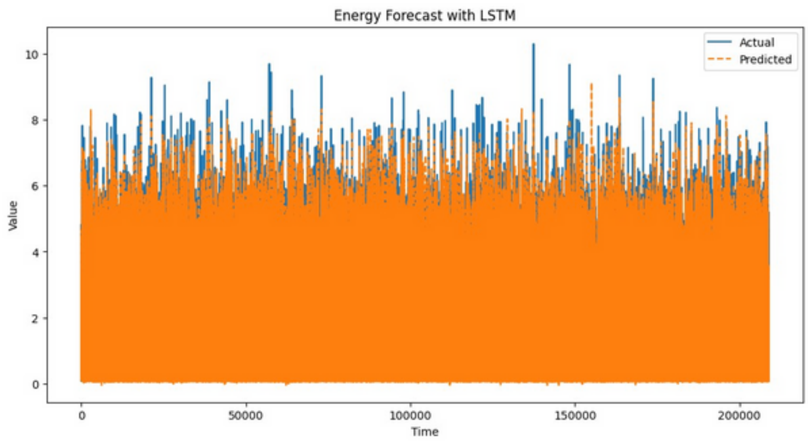
1. Data Cleaning and Preparation
    - Loaded CO_2 concentration, Energy consumption, and S&P 500 stock price datasets.
    - Cleaned data by converting date columns to datetime and handling missing values.
2. LSTM Modeling
    - Utilized LSTM neural networks for time series forecasting.
    - Data were prepared in sequences suitable for LSTM input.
    - Implemented LSTM models with dropout regularization for better generalization.
3. Model Training and Evaluation
    - Trained LSTM models on training data.
    - Evaluated models using Mean Squared Error (MSE) and Mean Absolute Error (MAE).
    - Plotted actual vs. predicted values for visual assessment.

Results with LSTM Model

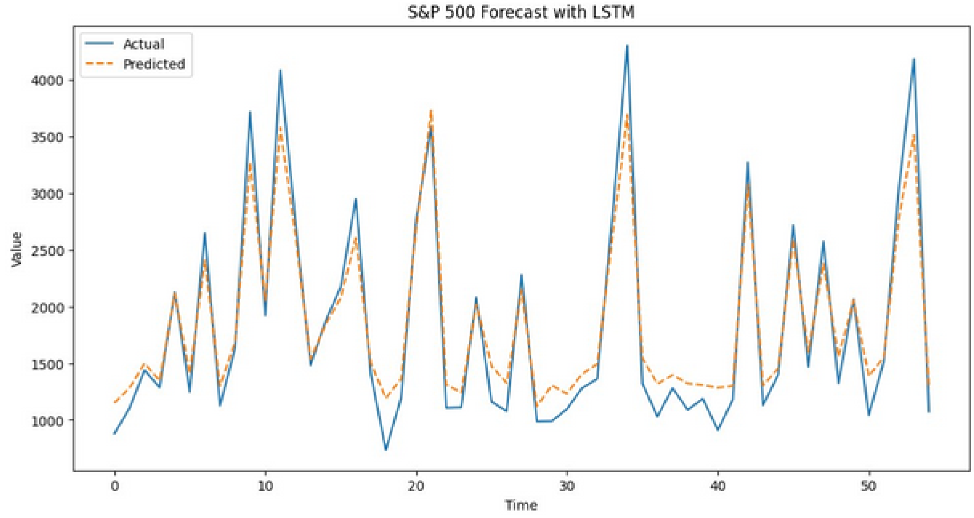
CO2 Dataset:



Energy Dataset:



S&P Dataset:

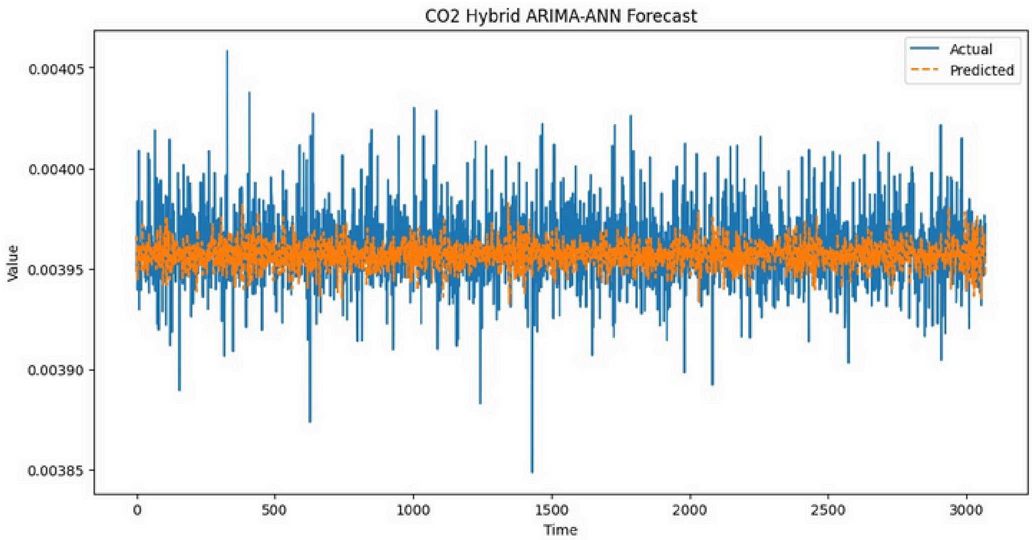


Hybrid Model

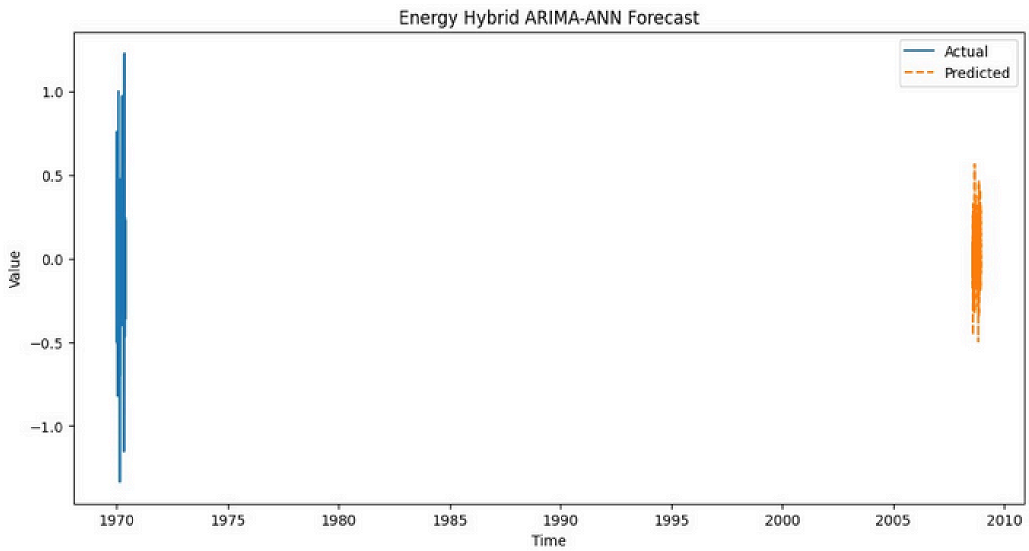
1. Data Preparation and Preprocessing
   - Loaded CO2 concentration dataset and performed Min-Max scaling.
   - Conducted log transformation and differencing to achieve stationarity.
   - Prepared differenced data for ARIMA modeling.
2. ARIMA Modeling
   - Utilized ARIMA model with order (5, 1, 0) for forecasting.
   - Trained ARIMA model on differenced CO2 data.
3. ANN Modeling for Residuals
   - Developed an Artificial Neural Network (ANN) to predict residuals from the ARIMA model.
   - Scaled ARIMA predictions and residuals for training.
4. Hybrid Model Integration
   - Combined ARIMA predictions with ANN predictions to obtain the final forecast.
   - Evaluated the hybrid model using Mean Squared Error (MSE) and Mean Absolute Error (MAE).
   - Plotted actual vs. predicted values for visual assessment.
   - Similarly Hybird model is adpoted for all datasets with different models

Results with Hybrid Model

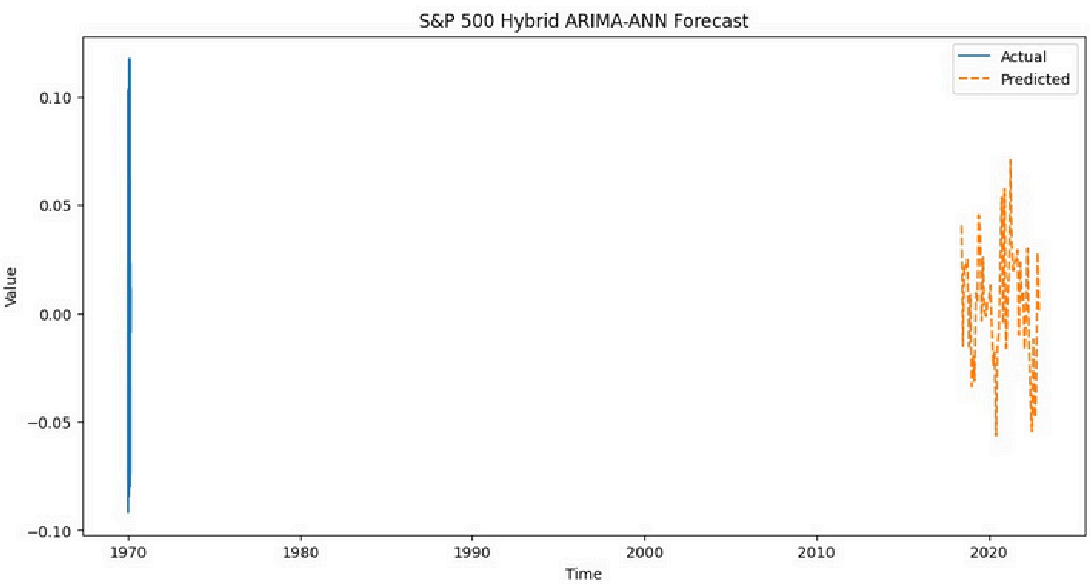
CO2 Dataset:



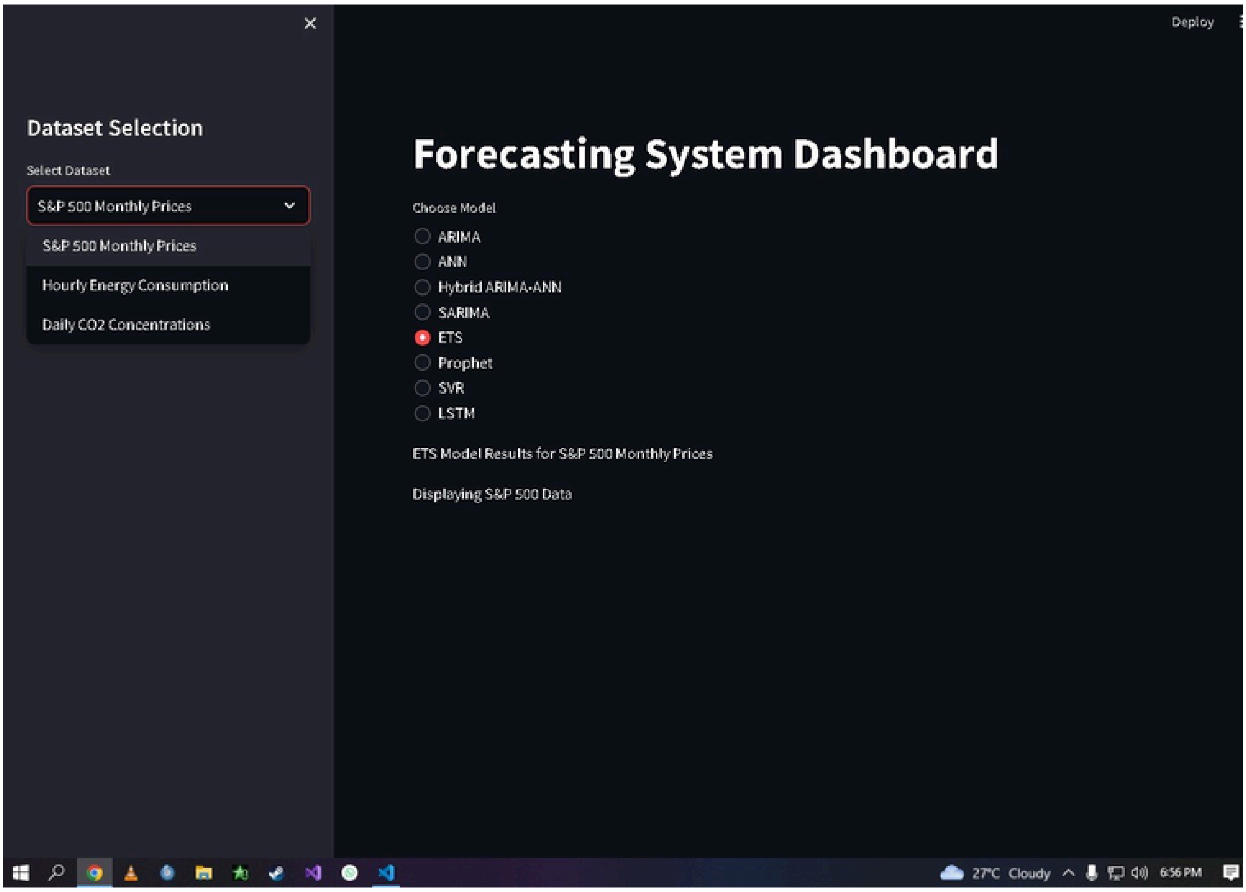
Energy Dataset:

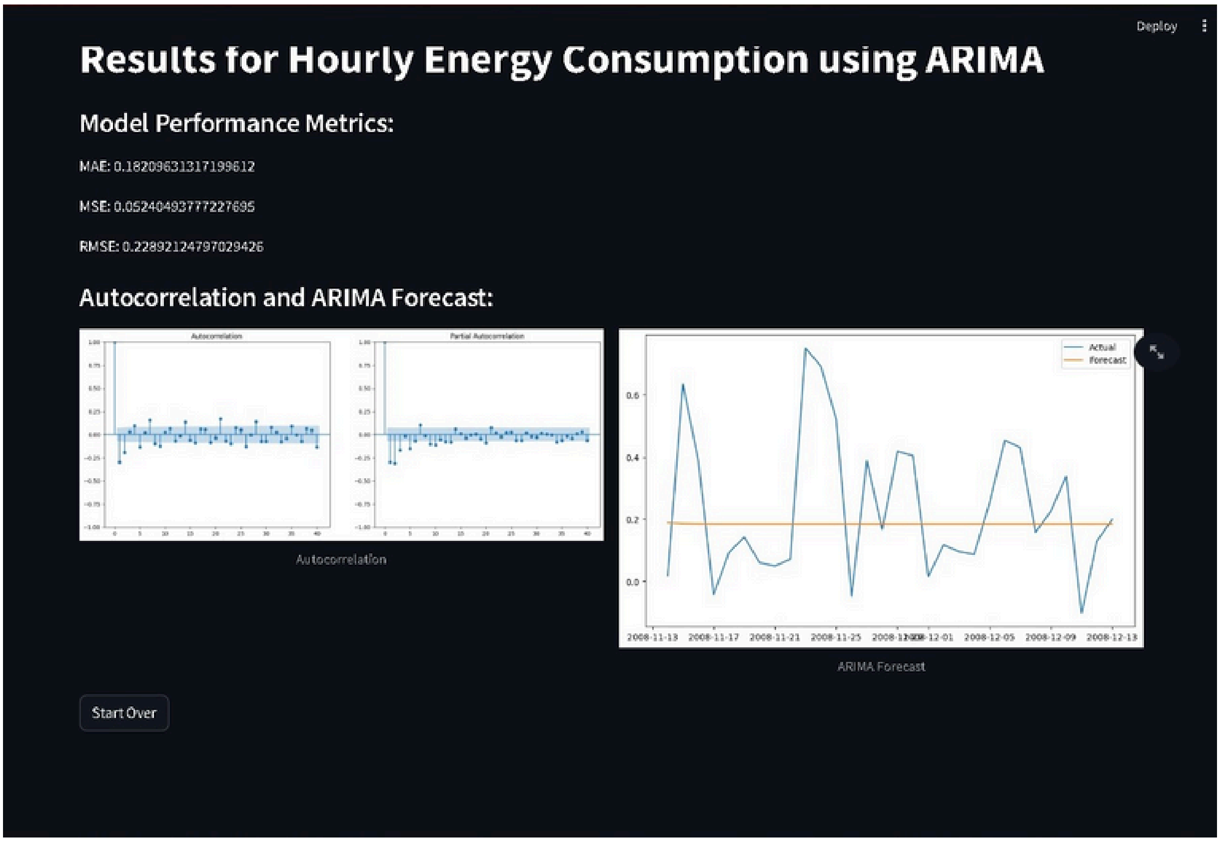


S&P Dataset:



Website


Deploy
Dataset Selection
Select Dataset
S&P 500 Monthly Prices
S&P 500 Monthly Prices
Hourly Energy Consumption
Daily CO2 Concentrations
Forecasting System Dashboard
Choose Model
ARIMA
ANN
Hybrid ARIMA-ANN
SARIMA
ETS
Prophet
SVR
LSTM
ETS Model Results for S&P 500 Monthly Prices
Displaying S&P 500 Data
27°C Cloudy
6:56 PM


Deploy
Results for Hourly Energy Consumption using ARIMA
Model Performance Metrics:
MAE: 0.18209631317199612
MSE: 0.05240493777227695
RMSE: 0.22892124797029426
Autocorrelation and ARIMA Forecast:
Autocorrelation
Partial Autocorrelation
Autocorrelation
Actual
Forecast
ARIMA Forecast
Start Over