PROJECT REPORT

Group members name:
Eisha Haroon (i212663)
Ayera Fatima (i212677)
Section: DS-M
Submitted to: Sir Saad Munir

Introduction

The objective of this project is to develop a comprehensive forecasting system that implements and compares different time series models across various sectors. The sectors in focus include Finance, Energy, and Environment. The models utilized for this project include ARIMA (Autoregressive Integrated Moving Average), ANN (Artificial Neural Network), Hybrid ARIMA-ANN, Exponential Smoothing, Prophet, and Support Vector Regression (SVR). The goal is to provide accurate forecasts for each sector by leveraging the strengths of these models and to present the results through a user-friendly interface.

To achieve this, we collected datasets specific to each sector: monthly stock prices for the Finance sector, hourly energy consumption data for the Energy sector, and daily atmospheric CO2 concentrations for the Environmental sector. The preprocessing steps involved data cleaning, normalization, and ensuring stationarity. Various models were then trained on the preprocessed data, tuned for optimal performance, and validated using historical data. The system was developed using a backend powered by Python and Flask, and a frontend built with ReactJS for dynamic interaction and visualization.

STEP 1: DATASET COLLECTION

Dataset Description:

1. Energy Sector Dataset

Source: Kagel

Entries: 145,366

Columns: 2 (Datetime, PJME_MW)

Details: This dataset contains hourly electricity load data for the PJM Interconnection (PJM) region. The "Datetime" column represents the timestamp of each observation, while "PJME_MW" indicates the electricity load in megawatts. The dataset spans from December 31, 2002, with a mean load of approximately 32,080 MW and a maximum load of 62,009 MW.

2. Environmental Sector Dataset

Source: Kagel

Entries: 5,572

Columns: 3 (Country, Year, CO2EmissionRate (mt))

Details: This dataset includes CO2 emission rates from various countries over several years. The "Country" column represents the name of the country, the "Year" column indicates the year of the observation, and the "CO2EmissionRate (mt)" column shows the CO2 emission rate in metric tons. The dataset spans from 1990 to 2021, providing comprehensive coverage of global CO2 emissions.

3. Financial Sector Dataset

Source: Kagel

Entries: 619,040

Columns: 7 (date, open, high, low, close, volume, Name)

Details: This dataset captures daily stock trading data for various companies. It includes the opening, highest, lowest, and closing prices, trading volume, and stock ticker symbol. The data spans multiple years, offering detailed insights into the stock market performance of numerous companies.

STEP 2: MODEL IMPLEMENTATIONS

Preprocessing steps: They were crucial to prepare the datasets for effective model implementation. These steps varied based on the requirements of each model and the nature of the data:

1. Data Cleaning: Removed or imputed missing values to ensure data completeness.

2. Normalization: Scaled numerical values to a standard range to improve model convergence and performance. Applied log transformations where necessary to stabilize variance and make patterns more discernible.

3. Ensuring Stationarity: Performed differencing on time series data to remove trends and seasonality, ensuring the data met the stationarity requirements for models like ARIMA and SARIMA.

4. Splitting Data: For large datasets, such as the financial sector data, split the data into smaller chunks to facilitate processing and model training.

MODELS:

1. ARIMA (AutoRegressive Integrated Moving Average):

Energy Dataset: Applied ARIMA to forecast future electricity load, leveraging historical load data patterns for accurate predictions.

Environmental Dataset: Used ARIMA to model and forecast CO2 emission trends over time, aiding in environmental policy planning.

Financial Dataset: Implemented ARIMA to predict future stock prices based on past price movements and trends.

2. SARIMA (Seasonal AutoRegressive Integrated Moving Average):

Energy Dataset: Utilized SARIMA to capture seasonal variations in electricity demand, enhancing the accuracy of load forecasting.

Environmental Dataset: Applied SARIMA to account for seasonal fluctuations in CO2 emissions, providing more precise forecasts.

Financial Dataset: Employed SARIMA to model and predict stock prices, considering seasonal effects and trends.

3. ANN (Artificial Neural Network) Design and Training:

Energy Dataset: Designed and trained ANN models to predict electricity load, learning complex patterns in the data for better accuracy.

Environmental Dataset: Used ANN to forecast CO2 emission rates, capturing non-linear relationships in the environmental data.

Financial Dataset: Implemented ANN to predict stock prices, leveraging the model's ability to learn intricate patterns in market data.

4. Exponential Smoothing:

Energy Dataset: Applied exponential smoothing techniques to smooth and forecast electricity load data, capturing short-term trends.

Environmental Dataset: Used exponential smoothing to model and predict CO2 emission trends, providing simple yet effective forecasts.

Financial Dataset: Employed exponential smoothing for stock price forecasting, focusing on capturing recent market trends.

5. Prophet:

Energy Dataset: Implemented Prophet to forecast electricity load, utilizing its capability to handle missing data and outliers effectively.

Environmental Dataset: Used Prophet for CO2 emission forecasting, benefiting from its robust handling of seasonal effects and trend changes.

Financial Dataset: Applied Prophet to predict stock prices, leveraging its strength in capturing daily seasonality and holiday effects.

6. Support Vector Regression (SVR):

Energy Dataset: Used SVR to predict electricity load, exploiting its ability to model complex relationships in the data.

Environmental Dataset: Implemented SVR to forecast CO2 emissions, taking advantage of its flexibility in handling non-linear data.

Financial Dataset: Applied SVR to predict stock prices, benefiting from its robustness in modeling financial time series data.

7. Long Short-Term Memory (LSTM):

Energy Dataset: Trained LSTM networks to forecast electricity load, utilizing its strength in learning long-term dependencies in time series.

Environmental Dataset: Used LSTM to model and predict CO2 emissions, capturing temporal dependencies and trends effectively.

Financial Dataset: Implemented LSTM to forecast stock prices, leveraging its ability to handle sequential and time-dependent data.

8. Hybrid Models Integration:

Energy Dataset: Combined ARIMA with LSTM to enhance electricity load forecasting by integrating linear and non-linear components.

Environmental Dataset: Integrated SARIMA and ANN models to improve CO2 emission forecasts, capturing both seasonal patterns and complex relationships.

Financial Dataset: Used a hybrid approach of Prophet and SVR to predict stock prices, blending the strengths of both models for better accuracy.

STEP 3: STORING VISUALIZATIONS IN SQL LITE

To efficiently manage and retrieve visualizations generated from the models, the following steps were undertaken:

Database Creation: Established a SQLite database to store visualizations, ensuring easy access and management.

Table Structure: Designed tables to hold various types of visualizations with metadata (model type, dataset, timestamp).

Data Insertion: Stored visualizations as binary data, associating them with relevant metadata for quick retrieval.

Query and Retrieval: Implemented SQL queries for efficient visualization retrieval based on criteria like model type, date range, or dataset.

STEP 4: CONNECTING THE DATABASE TO THE FRONT END THROUGH FLASK

To provide a seamless and interactive user experience, I connected the SQLite database containing the visualizations to the front-end interface using Flask, a lightweight web framework for Python. Here's how this was accomplished:

Flask Setup:

- Installed Flask and set up a basic Flask application to serve as the backbone of the web interface.
- Configured the Flask app to connect to the SQLite database, enabling data retrieval and interaction.

Database Connection:

- Established a connection to the SQLite database within the Flask application.

Endpoints Creation:

- Created RESTful API endpoints to serve the visualizations and forecast data to the front-end. These endpoints handle requests for different types of visualizations and model outputs.
- Each endpoint returns data in JSON format, ensuring compatibility and ease of integration with the front-end.

Front-End Integration:

- Developed a user-friendly front-end interface using HTML, CSS, and JavaScript, integrating data visualization libraries such as D3.js and Plotly for dynamic and interactive charts.
- Connected the front-end to the Flask back-end, allowing users to interact with the visualizations, select different models, and view forecasts directly from the web interface.

Challenges and Solutions:

1. Large Dataset Handling:

Challenge: The financial sector dataset was too large to process efficiently in a single run.

Solution: We divided the dataset into smaller, manageable chunks to facilitate processing and model training.

2. Integration Difficulties:

Challenge: Integrating the front-end with the SQLite database presented challenges.

Conclusion:

In this project we developed a forecasting system that leverages multiple time series models to provide accurate predictions across the Finance, Energy, and Environment sectors. The integration with a user-friendly interface makes it a valuable tool for analysts and decision-makers.