HALDIA INSTITUTE OF TECHNOLOGY

MACHINE LEARNING PROJECT

Arijit Dutta, Soham Chowdhury, Sneha Singh,
Sneha Chourasia, Brijit Maity, Arighna Jha,
Dept of Electronics and Communication Engineering
(Haldia Institute of Technology)

***Abstract*- This paper presents a comparative analysis of three different machine learning techniques—Prophet, Stats models, and KNIME—applied to three distinct datasets for time series forecasting. The study evaluates the accuracy, computational efficiency, and suitability of each method for various forecasting applications. The results demonstrate the strengths and limitations of each approach, providing insights into their practical implementations. This paper presents a comparative study of three different forecasting techniques: Facebook Prophet, Stats model (ARIMA and Exponential Smoothing), and KNIME. The study evaluates their performance on three distinct datasets, each representing different characteristics and forecasting challenges. Prophet, developed by Facebook, is a robust tool designed to handle missing values and seasonality with minimal parameter tuning. This study offers insights into selecting the most suitable forecasting technique based on dataset characteristics, computational complexity, and ease of implementation**

I. Project Overview

Overview The Machine learning Project is designed with a clear and ambitious scope that aims to transform the way organizations manage their workforce. By leveraging full stack technologies, the EMS seeks to achieve the following objectives:

- The **first** dataset involves Advertising Budget and Sales Prediction, where Stats models is employed to analyze the relationship between advertising expenditures and sales. Using Ordinary Least Squares (OLS) regression, we assess the impact of different advertising channels on sales performance and forecast future sales based on budget allocations.
- The **second** dataset focuses on Airline Passenger Profit Prediction, where Facebook Prophet is used to analyze past airline passenger traffic and predict future profitability. Prophet, a robust forecasting tool developed by Facebook, effectively handles seasonality, trends, and missing data, making it suitable for predicting future revenue based on anticipated passenger growth.
- **Third** dataset examines Country GDP Forecasting, utilizing KNIME with Linear and Polynomial Regression models. By leveraging KNIME's no-code/low-code machine learning workflow, we develop predictive models to estimate future GDP based on historical economic indicators. Linear regression provides a straightforward trend analysis, while polynomial regression captures non-linear economic fluctuations, offering a more flexible approach to GDP forecasting.

II.Key Features

The EMS differentiates itself from other employee management systems through several innovative features:

- **Comparative Analysis of Forecasting Methods**: Evaluates three different forecasting techniques: Stats models (OLS Regression), Facebook Prophet, and KNIME (Linear & Polynomial Regression). Analyzes their suitability for different datasets and forecasting applications.
- **Diverse Datasets for Real-World Applications**: Advertising Budget and Sales Prediction using Stats models (OLS Regression) to determine the impact of advertising on sales. Airline Passenger Profit Forecasting using Facebook Prophet to predict future airline revenues based on passenger trends. Country GDP Forecasting using KNIME's Linear and Polynomial Regression for economic analysis..
- **Evaluation with Standard Error Metrics :** Performance comparison based on Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE).
- **Future Scope & Research Direction**: Proposes the integration of deep learning models (LSTMs) and hybrid forecasting approaches.

III. Objective

The primary objective of this report is to analyze and compare the effectiveness of three different forecasting techniques—Stats models, Facebook Prophet, and KNIME—applied to three distinct datasets. The study aims to provide insights into the suitability of each method for different time series forecasting tasks.

Specific Objectives:

- **Evaluate the Performance of Different Forecasting Models**: Assess the accuracy,

efficiency, and ease of implementation of Stats models (OLS Regression), Prophet, and KNIME (Linear & Polynomial Regression).

- **Analyze the Impact of Advertising Budgets on Sales Prediction**: Use Stats models (OLS Regression) to study how different advertising channels influence sales and forecast future sales based on budget allocations.
- **Predict Airline Passenger Profit Trends:** Implement Facebook Prophet to analyze past airline passenger data and forecast future profitability based on seasonal trends and growth patterns.
- **Compare Models Based on Standard Evaluation Metrics:** Measure and compare forecasting accuracy using Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE).
- **Identify the Strengths and Limitations of Each Method:** Discuss the advantages and drawbacks of each forecasting approach based on dataset characteristics and forecasting complexity.
- **Provide Insights for Practical Applications:** Help businesses, analysts, and researchers choose the most suitable forecasting method based on their data and objectives.
- **Explore Future Enhancements in Forecasting Techniques:** Suggest improvements such as deep learning models (LSTMs), hybrid approaches, and automated machine learning (AutoML) for more accurate and scalable forecasting solutions.

IV. Models Used

1. Linear Regresssion

Linear regression models the relationship between a dependent variable and one or more independent variables using a linear equation. It is implemented in **StatsModels** using OLS() (Ordinary Least Squares), which estimates coefficients by minimizing the sum of squared residuals.

2. Polynomial Regression

Polynomial regression extends linear regression by introducing polynomial terms of the independent variable. It is useful for capturing non-linear relationships in data. This is implemented using numpy.polyfit() or by creating polynomial features with **StatsModels**.

3. OLS Regression (Ordinary Least Squares)

OLS regression is a fundamental method for estimating unknown parameters in a linear regression model. It minimizes the sum of squared residuals and is implemented in **StatsModels** using:

```
import statsmodels.api as sm
X = sm.add_constant(X)
model = sm.OLS(y, X).fit()
print(model.summary())
```

4. EWMA (Exponentially Weighted Moving Average)

EWMA is used for smoothing time series data by applying exponentially decreasing weights to past observations. It gives more importance to recent data points, making it suitable for financial and economic applications.

V. Datasets

Three different datasets are used to assess the performance of the forecasting models

- **Advertising Budget and Sales Prediction**: This dataset contains advertising expenditure data across different media channels and corresponding sales figures. The goal is to predict future sales based on budget allocation.

- **Airline Passenger Seasonal Forecasting**: This dataset records historical airline passenger counts. Using this data, we forecast future passenger traffic and estimate potential profits.

- **Adult population database :** The dataset used in this study is the Adult Population Income dataset, obtained from the UCI Machine Learning Repository. The dataset consists of 48,842 records with 14 attributes, including age, workclass, education level, marital status, occupation, relationship, race, sex, capital gain, capital loss, hours per week, and native country. The target variable is income, which is categorized into two classes: <=50K and >50K.

VI. Data Cleaning and Preprocessing

Before applying forecasting models, data cleaning and preprocessing are essential to improve accuracy and ensure reliable predictions. The following steps were performed on all datasets:

- **Handling Missing Values**: Missing values were either imputed using statistical methods (e.g., mean/median imputation) or removed if they were insignificant.

- **Outlier Detection and Removal**: Outliers were identified using Z-score analysis and IQR (Interquartile Range) methods, and necessary adjustments were made to maintain data integrity.

- **Data Normalization and Scaling**: Feature scaling techniques such as Min-Max Scaling and Standardization were applied to ensure model consistency.

- Date Formatting and Indexing: For time series forecasting, date columns were converted into proper datetime format and set as the index.

- Feature Engineering: New features such as lag variables, rolling averages, and seasonality indicators were created to enhance model performance.

VIII. Advertising Budget and Sales Prediction (statsmodel)

Step 1: Data Collection:
Load the dataset containing advertising expenditures (TV, radio, newspapers) and corresponding sales.
Step 2: Data Preprocessing:
Handle missing values by filling them with appropriate methods.
Detect and remove outliers using statistical techniques like score or IQR.
Normalize or standardize the data for consistency.
Step 3: Feature Selection:
Identify independent variables (TV, radio, newspaper budgets) and dependent variable (sales).
Step 4: Model Training (Statsmodels - OLS Regression):
Use Ordinary Least Squares (OLS) regression to analyze relationships between ad budgets and sales.
Fit the model to the training dataset.
Step 5: Prediction:
Provide new advertising budget data as input.
The model generates predicted sales based on the regression equation.
Step 6: Evaluation:
Assess performance using R-squared, RMSE, and MAE metrics.
Interpret coefficients to understand the influence of each ad channel. Tools Used: JMeter, Apache Benchmark (AB), LoadRunner.

VII. Airline Passenger Profit Forecasting (Prophet)

Step 1: Data Collection:

Load historical airline passenger data with date and passenger count.

Step 2: Data Preprocessing:

Convert the dataset into a time-series format.Handle missing values using interpolation techniques.Ensure date columns are in proper datetime format.

Step 3: Model Training (Facebook Prophet):

Define the time-series model with seasonality, trend, and holidays incorporated.Train the Prophet model using historical passenger data

Step 4: Future Prediction:

Set a forecast horizon (e.g., next 12 months).Generate predicted passenger numbers based on learned trends.

Step 5: Profit Estimation:

Multiply predicted passenger count with estimated revenue per passenger.Forecast potential profit for upcoming months/years.

Step 6: Evaluation:

Use MAPE, RMSE, and MAE to compare predicted vs. actual values.Visualize forecast trends to check for seasonality effects.

IX. Adult Population Income dataset

Step 1: Data Import

- Load the dataset using the **File Reader** or **CSV Reader** node.
- Verify the structure of the dataset to ensure all attributes are correctly imported.

Step 2: Data Preprocessing

- **Handling Missing Values:**
- Use the **Missing Value** node to replace missing categorical attributes with the most frequent values.

- **Encoding Categorical Variables:**
  - Apply the **One-Hot Encoding** node for categorical features or use **Category to Number** to label encode categorical variables.
- **Data Splitting:**
  - Utilize the **Partitioning** node to split data into **80% training** and **20% testing** subsets.
- **Feature Scaling:**
  - Apply the **Normalizer** node to normalize numerical attributes (e.g., age, hours per week) using Min-Max scaling.

Step 3: Decision Tree Model Implementation

- **Model Training:**
  - Use the **Decision Tree Learner** node with Gini impurity as the splitting criterion.
- **Prediction:**
  - Apply the trained model to the test set using the **Decision Tree Predictor** node.
- **Performance Evaluation:**
  - Use the **Scorer** node to evaluate the model's accuracy, precision, recall, and F1-score.

Step 4: Model Performance Analysis

- Display model results using the **Confusion Matrix** node.
- Evaluate accuracy, precision, recall, and F1-score.
- Compare Decision Tree performance with other models (e.g., Logistic Regression, Random Forest)

using **Multilayer Perceptron Learner** or **Random Forest Learner** nodes.

Step 5: Feature Importance Analysis

- Use the **Decision Tree View** node to analyze feature importance.
- Identify the most significant attributes influencing income classification (e.g., education level, capital gain, hours per week).

Step 6: Decision Tree Visualization

- Use the **Tree Viewer** node to visualize the tree structure.
- Optimize tree depth to prevent overfitting.

Step 7: Comparative Analysis

- Compare Decision Tree results with other models using the **ROC Curve** and **Lift Chart** nodes.

Step 8: Limitations and Future Work

- Analyze overfitting tendencies using **Cross-Validation** node.
- Remove low predictive power attributes to refine the model.
- Implement advanced ensemble methods (e.g., Gradient Boosting) using **Gradient Boosted Trees Learner** for performance improvements.

X. Exponentially Weighted Moving Average (EWMA)

The purpose of this analysis is to understand weather patterns by analyzing rainfall data and predicting future values. EWMS provides a way to smooth fluctuations in the dataset by assigning greater importance to recent observations. The rolling mean method allows us to observe trends over a defined window period, which helps in making short-term predictions. By visualizing these methods, we can better interpret historical weather trends and gain insights into future conditions. This document includes data preprocessing, calculations, and visualizations to demonstrate how these techniques can be applied effectively in data analysis.

2. Libraries Used

- **pandas**: For data handling and manipulation
- **numpy**: For numerical operations
- **matplotlib.pyplot**: For visualization of results

Dataset Overview

- **Source**: The dataset is obtained from a weather monitoring system and stored in a CSV file.
- **Number of Columns**: The dataset consists of 22 columns covering various meteorological parameters.
- **Key Features**:
  - Location: Name of the city or region where the data is collected.
  - MinTemp: Minimum temperature recorded in degrees Celsius.
  - MaxTemp: Maximum temperature recorded in degrees Celsius.
  - Rainfall: The amount of rainfall recorded in millimeters.
  - Evaporation: Total evaporation observed in millimeters.
  - Sunshine: Number of hours of sunshine in a day.
  - WindGustDir: Direction of the strongest wind gust.
  - WindGustSpeed: Speed of the strongest wind gust in km/h.
  - Humidity9am & Humidity3pm: Humidity levels recorded in the morning and afternoon.
  - Pressure9am & Pressure3pm: Atmospheric pressure recorded at different times of the day.
  - Temp9am & Temp3pm: Temperature readings taken at 9 AM and 3 PM.
  - RainToday: A categorical feature indicating whether it rained that day (Yes/No).
- **Data Period**: The dataset contains daily weather records spanning multiple years.
- **Missing Data**: Some columns contain missing values, particularly in Evaporation and Sunshine.
- **Primary Focus**: This analysis primarily focuses on the Rainfall column to predict future trends using EWMS and Mean calculations.

3. Libraries Used

- **pandas**: For data manipulation and analysis.
- **numpy**: For numerical operations.
- **matplotlib.pyplot**: For visualization.

Introduction

This document provides a step-by-step explanation of how to compute the Exponential Weighted Moving Sum (EWMS) and Mean, along with their graphical representation and predictions using Python in Google Colab.

1. Importing Necessary Libraries

To begin, we import the required libraries:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

- **pandas**: Used for data manipulation and reading CSV files.

- **numpy**: Used for numerical operations.
- **matplotlib.pyplot**: Used for plotting graphs.

2. Loading the Dataset

The dataset is loaded using pandas:

```
file_path = "Weather Test Data.csv"
df = pd.read_csv(file_path)
display(df.head())
```

This reads the CSV file and displays the first few rows to understand the data structure.

3. Selecting the Column for Analysis

We select the **'Rainfall'** column for performing EWMS and Mean calculations:

```
column_name = 'Rainfall'
```

4. Applying EWMS and Mean Calculation

If the selected column exists, we apply the Exponential Weighted Moving Sum and calculate the mean:

```
if column_name in df.columns:
    # EWMS Calculation
    df['EWMS'] = df[column_name].ewm(span=5,
      adjust=False).mean()

    # Mean Calculation
    df['Mean'] =
      df[column_name].rolling(window=5).mean()

    # Prediction using Mean (Rolling Forecast)
    df['Prediction'] = df['Mean'].shift(-1)
```

- **EWMS**: Provides a weighted moving average where recent values have higher significance.
- **Mean**: Calculates the simple moving average over a given window.
- **Prediction**: Shifts the mean values to project the next day's value.

5. Visualizing the Results

The results are plotted using Matplotlib:

```
    plt.figure(figsize=(12, 6))
    plt.plot(df.index[:365], df['EWMS'][:365],
      label='EWMS', color='blue')
    plt.plot(df.index[:365], df['Mean'][:365], label='Mean',
      color='red')
    plt.plot(df.index[:365], df['Prediction'][:365],
      label='Prediction', color='green', linestyle='dashed')
    plt.xlabel('Days')
    plt.ylabel('Values')
    plt.title('EWMS, Mean, and Prediction for 1 Year')
    plt.legend()
    plt.show()
```

- The **EWMS** is shown in blue.
- The **Mean** is displayed in red.
- The **Predicted Values** are represented as a dashed green line.

6. Displaying the Results

The calculated values are displayed as a table:

```
display(df[['row ID', column_name, 'EWMS', 'Mean',
    'Prediction']].head(10))
```

XI.ACKNOWLEDGMENTS

The authors would like to express their sincere gratitude to all individuals and organizations who contributed to the successful completion of this project. Special thanks to our mentors, colleagues, and academic advisors for their invaluable guidance and feedback. We also acknowledge the resources and support provided by the research institutions and technological communities that made this study possible.

XII. REFRENCES

[1] J. D. Hamilton, *Time Series Analysis*, Princeton University Press, 1994. DOI: 10.1515/9780691218632.

[2] R. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 2nd ed., OTexts, 2018. DOI: 10.32614/CRAN.package.fpp2.

[3] J. Perktold, S. Seabold, and J. Taylor, “Statsmodels: Econometric and statistical modeling with Python,” *Proc. 9th Python in Science Conference (SciPy)*, 2010. DOI: 10.25080/Majora-92bf1922-011.

[4] S. J. Taylor and B. Letham, “Forecasting at scale,” *The American Statistician*, vol. 72, no. 1, pp. 37–45, 2018. DOI: 10.1080/00031305.2017.1380080.

[5] M. Berthold et al., “KNIME: The Konstanz Information Miner,” in *Studies in Classification, Data Analysis, and Knowledge Organization*, Springer, 2007. DOI: 10.1007/978-3-540-78246-9_38.

[6] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*, 5th ed., Wiley, 2012. DOI: 10.1002/9781118135159.

[7] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed., Springer, 2009. DOI: 10.1007/978-0-387-84858-7.

[8] Python Software Foundation, “Python 3.8 Documentation.” Available: https://docs.python.org/3.8/..