Machine Learning: EDA, Data Mining, Time Series Forecast and Knime Analytics

This presentation outlines the use of machine learning techniques for analyzing and forecasting Gross Domestic Product (GDP). We will explore various methodologies, from data mining and exploratory data analysis (EDA and advanced time series modeling with Meta Prophet and KNIME Analytics Platform. Our goal is to provide a comprehensive understanding of how machine learning can be applied to economic forecasting, offering valuable insights for policymakers and financial analysts.

Team Members

Leader: Arijit Dutta (22/ECE/054)

Members:

Arijit Dutta (22/ECE/054)
Arighna Jha(22/ECE/053)
Brijit Maity(22/CSE-CS/025)
Sneha Chourasia(22/ECE/169)
Sneha Singh(22/ECE/170)
Soham Chowdhury(22/ECE/172)

Project Overview: Goals, Data Sources, and Tools

Goals

- Accurate GDP Forecasting: Predict future GDP values with high precision.
- Insightful Analysis: Identify key economic indicators and their impact on GDP.
- Tool Proficiency: Master Meta Prophet and KNIME for time series analysis.

Data Sources

- EDA: 1. Economies of Scale Data
  2. Employee Salary Dataset
- Prophet : Airline Passengers monthly travel data.
- Knime Analytics: Adult Population Dataset

Tools

- Colab (Python): Data manipulation, analysis, and modeling.
- Meta Prophet: Time series forecasting.
- KNIME: Data integration, workflow automation, and visualization.

Exploratory Data Analysis (EDA): Unveiling Key Insights

Data Cleaning

Handling missing values and outliers to ensure data quality. We will impute missing data using statistical methods and remove or transform outliers using techniques like winsorization.

Descriptive Statistics

Calculating mean, median, standard deviation, and other key metrics. These statistics will provide a summary of the central tendency and variability of the GDP data.

Visualizations

Creating histograms, scatter plots, and time series plots to identify patterns. We will use these plots to understand the distribution of GDP values and their trends over time.

Model Summary

Linear and Polynomial Regression

- **Linear Regression** fits a **straight-line relationship** between independent (**Number of Units**) and dependent (**Manufacturing Cost**) variables, following the equation **y=mx+by**.
- **Polynomial Regression** extends Linear Regression by introducing higher-degree terms ($ax^2 + bx + c = y$)capturing **non-linear trends** in data.

Project Insight: The model started with **Linear Regression**, but **Polynomial Regression (degree = 2)** was applied to check for **non-linear dependencies in cost prediction**.

Statistics Model

- A comprehensive Python library offering classes and functions for estimating various statistical models, conducting tests, and performing data exploration. It provides detailed result statistics for each estimator, ensuring accuracy by validating results against established statistical packages.
- **Application in Our Project:**

Model Evaluation: Leveraged the comprehensive summary reports from statsmodels to assess model performance, including metrics like R-squared, F-statistic, and p-values, facilitating informed decision-making.

Meta Prophet

- Developed by Meta, Prophet is an open-source forecasting tool designed for time series data.
- It models data using an additive approach, capturing non-linear trends with components for yearly, weekly, and daily seasonality, as well as holiday effects.

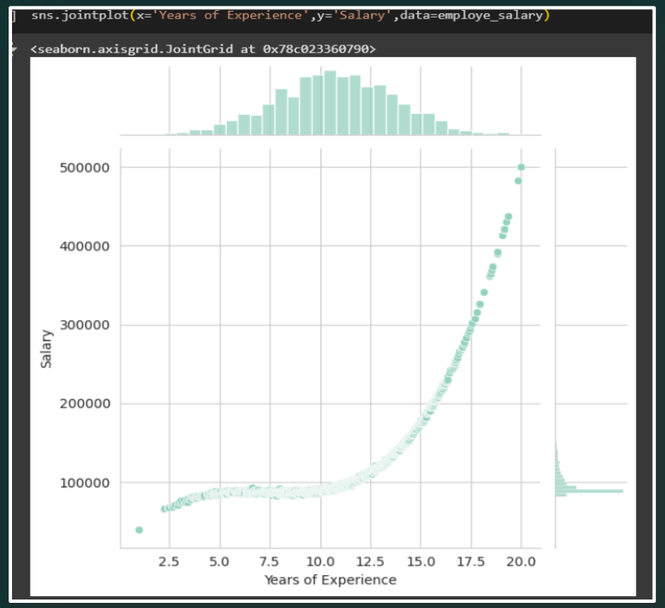
Application in Our Project:

•Utilized Prophet to forecast manufacturing costs over time, accounting for potential seasonal variations and trend changes.

All Analytics at a Glance

```
sns.jointplot(x='Years of Experience',y='Salary',data=employe_salary)
```

```
<seaborn.axisgrid.JointGrid at 0x78c023360790>
```



Linear Regression

```
print(model.summary())
```

OLS Regression Results

Dep. Variable:	Sales ($)	R-squared:	0.612
Model:	OLS	Adj. R-squared:	0.610
Method:	Least Squares	F-statistic:	312.1
Date:	Wed, 12 Feb 2025	Prob (F-statistic):	1.47e-42
Time:	18:38:31	Log-Likelihood:	-519.05
No. Observations:	200	AIC:	1042.
Df Residuals:	198	BIC:	1049.
Df Model:	1		
Covariance Type:	nonrobust		

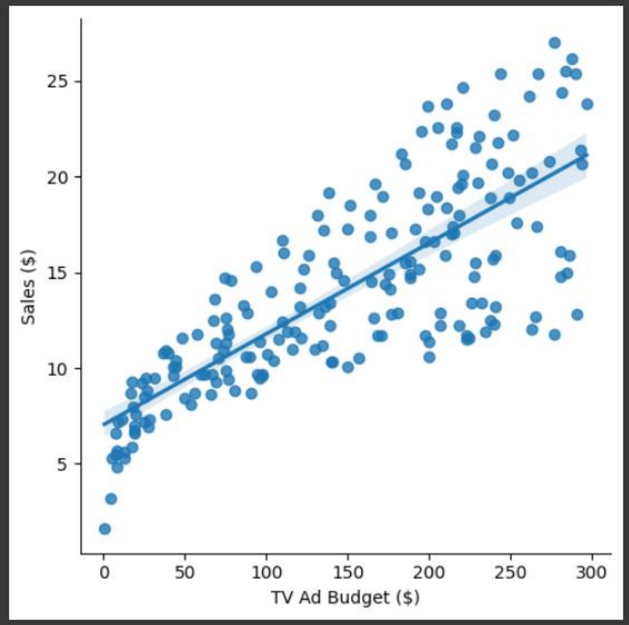
	coef	std err	t	P>|t|	[0.025	0.975]
const	7.0326	0.458	15.360	0.000	6.130	7.935
TV Ad Budget ($)	0.0475	0.003	17.668	0.000	0.042	0.053

Omnibus:	0.531	Durbin-Watson:	1.935
Prob(Omnibus):	0.767	Jarque-Bera (JB):	0.669
Skew:	-0.089	Prob(JB):	0.716
Kurtosis:	2.779	Cond. No.	338.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Satistics Model

```
# Scatterplot of TV vs. Sales
sns.lmplot(x="TV Ad Budget ($)", y="Sales ($)", data=data)
plt.show()
```

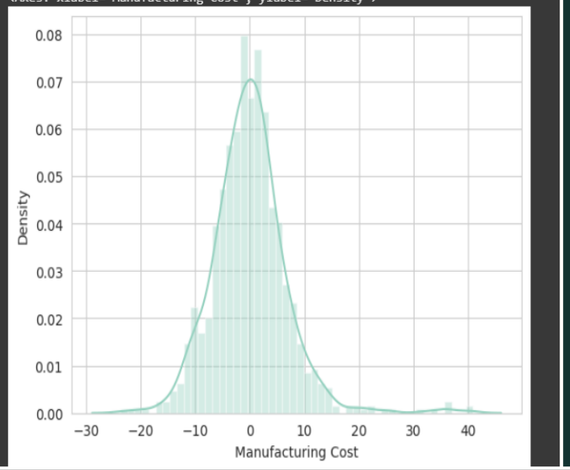


The scatter plot shows a positive linear relationship between TV ad budget and sales, but the presence of outliers suggests that other factors may also influence sales.

OLS Statistical Model

```
sns.distplot((y_train-predections),bins=50)
```

```
<Axes: xlabel='Manufacturing Cost', ylabel='Density'>
```

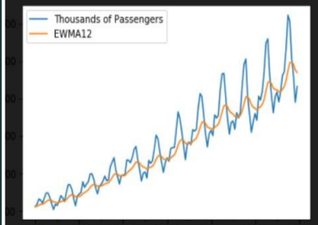


so we can find out that the deta was correct because when the number of units increase then the manufacture price also decrease

Polynomial Regression

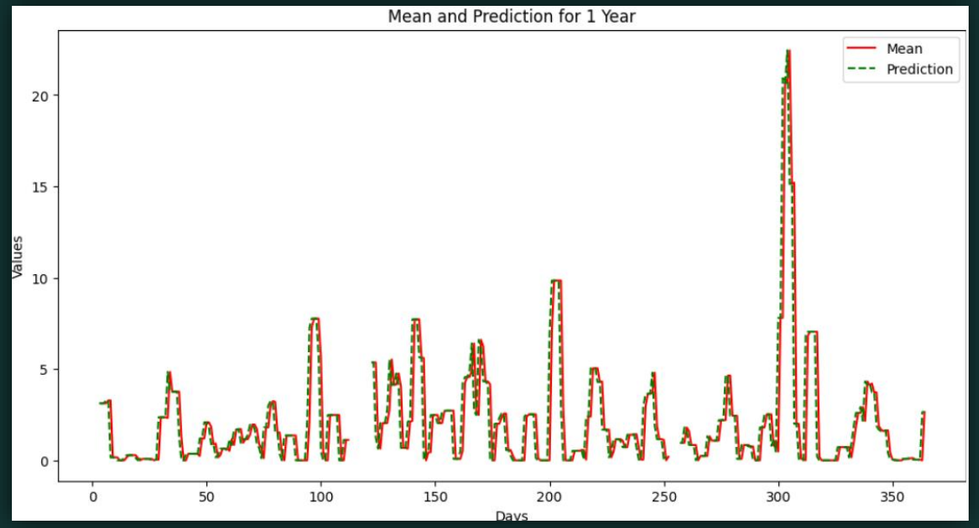
```
airline['EWMA12'] = airline['Thousands of Passengers'].ewm(span=12,adjust=False).mean()
```

```
airline[['Thousands of Passengers','EWMA12']].plot();
```



EWMA {Exponentially Weighted Moving Average} (EDA)

Exponentially Weighted Moving Average(EWMA): in Feature Engineering and Selection




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

	row ID	Rainfall	Mean	Prediction
0	Row0	0.0	NaN	NaN
1	Row1	0.0	NaN	NaN
2	Row2	0.0	NaN	NaN
3	Row3	0.0	NaN	3.12
4	Row4	15.6	3.12	3.12
5	Row6	0.0	3.12	3.12
6	Row7	0.0	3.12	3.28
7	Row8	0.8	3.28	3.28
8	Row9	0.0	3.28	0.16
9	Row10	0.0	0.16	0.16

so this is the predection in weather

1

Smoothing Temporal Data:

EWMA is widely used to smooth time-series data by giving higher weight to recent observations while retaining historical trends, reducing noise and enhancing trend detection in predictive models.

2

Feature Importance in Anomaly Detection:

By applying EWMA to features, deviations from expected values can be identified, making it useful for detecting anomalies, drifts, or rare events in data streams.

3

Dimensionality Reduction Aid

EWMA-transformed features can be used to identify the most stable and predictive attributes over time, supporting feature selection by reducing redundant or less informative features.

Forecasting Airline Passenger Trends Using Meta Prophet: A Time Series Analysis

Data Preparation

Formatting the data for Meta Prophet, including creating 'ds' (date) and 'y' (No. of Passengers) columns. Ensure the data is structured in a way that the model can easily understand.

Model Training

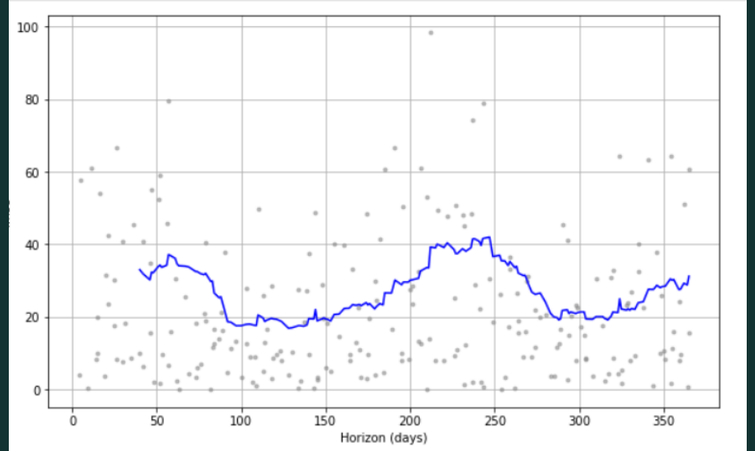
Training the Meta Prophet model can be done on historical psssenger data. This involves fitting the model to the time series data and adjusting parameters to optimize performance.

Forecasting

Generating future Seasonality forecasts on passenger travels using the trained model. Extend the model into the future to predict booking and travels plans ahead for upcoming periods.

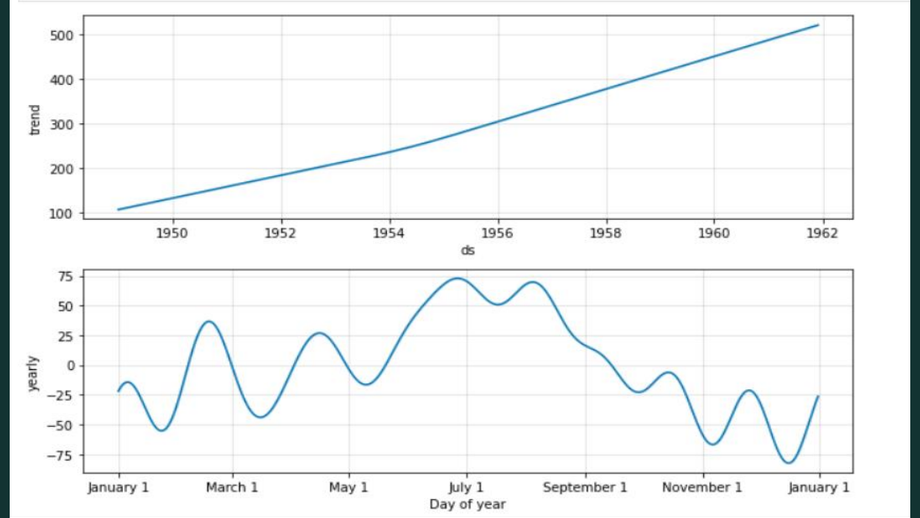
Prophet Dashboard

```
from fbprophet.plot import plot_cross_validation_metric
fig = plot_cross_validation_metric(df_cv, metric='rmse')
```



Cross Validation Metrics

```
#### Visualize Each Components[Trends,yearly]
model.plot_components(prediction)
```



Trends and Yearly Visualisation

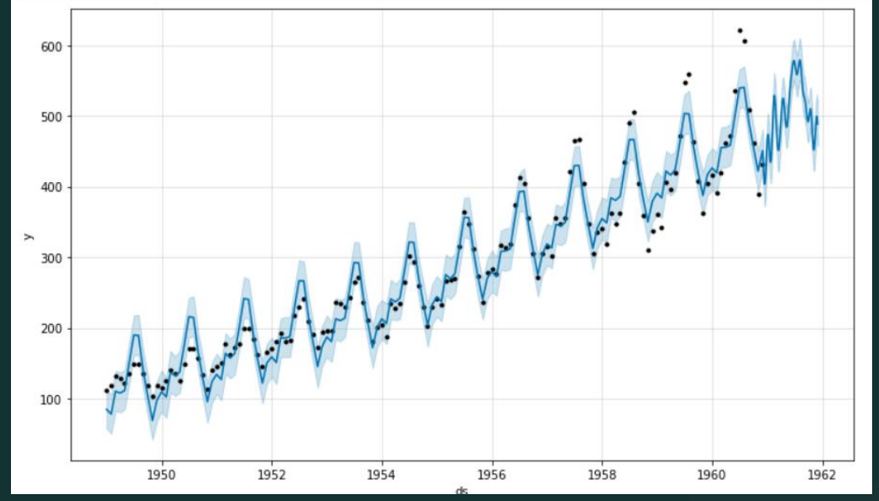
```
prediction=model.predict(future_dates)
```

```
prediction.head()
```

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additive_terms_lower	additive_terms_upper	
0	1949-01-01	106.583811	57.631562	111.708136	106.583811	106.583811	-21.946575	-21.946575	-21.946575	-21
1	1949-02-01	108.760063	50.452764	105.815553	108.760063	108.760063	-30.707281	-30.707281	-30.707281	-30
2	1949-03-01	110.725710	82.783039	139.039762	110.725710	110.725710	-0.469476	-0.469476	-0.469476	-0
3	1949-04-01	112.901962	80.296089	136.975932	112.901962	112.901962	-5.166670	-5.166670	-5.166670	-5
4	1949-05-01	115.008012	84.868864	140.400938	115.008012	115.008012	-3.765920	-3.765920	-3.765920	-3

Prediction Output: one year ahead.

```
### plot the predicted projection
model.plot(prediction)
```



Prediction Output Plot

Introduction to KNIME Platform Data Analytics

Overview

1

KNIME is an open-source platform for data analytics, reporting, and integration. It provides a visual workflow environment for designing and executing data science tasks.

Key Features

2

- Drag-and-drop interface.
- Extensive node library.
- Integration with various data sources.

Use Cases

3

- Data integration and transformation.
- Predictive modeling.
- Business intelligence and reporting.

KNIME Workflow: Data Integration and Transformation

1. Data Import & Cleaning

- **Data Source**: Utilizes the Adult dataset from the UCI Machine Learning Repository.
- **Import Method**: Reads data using the File Reader node in KNIME.
- **Data Cleaning Steps**:

•Handles missing values by replacing them with the mode or median.
•Removes irrelevant columns to streamline the dataset.
•Filters out records with inconsistent or erroneous entries.

2. Data Information & Transformation

•**Data Overview**:
•Features include age, education, occupation, and hours per week.
•Target variable: income category (<=50K or >50K).
•**Transformations Applied**:
•Encodes categorical variables using one-hot encoding.
•Normalizes numerical features to a standard scale.
•Generates new features, such as age groups, to enhance model performance.

3. Insights & Forecasting

- **Modeling Approach**:

•Employs a Decision Tree classifier to predict income categories.
•Splits data into training and testing sets to evaluate model performance.

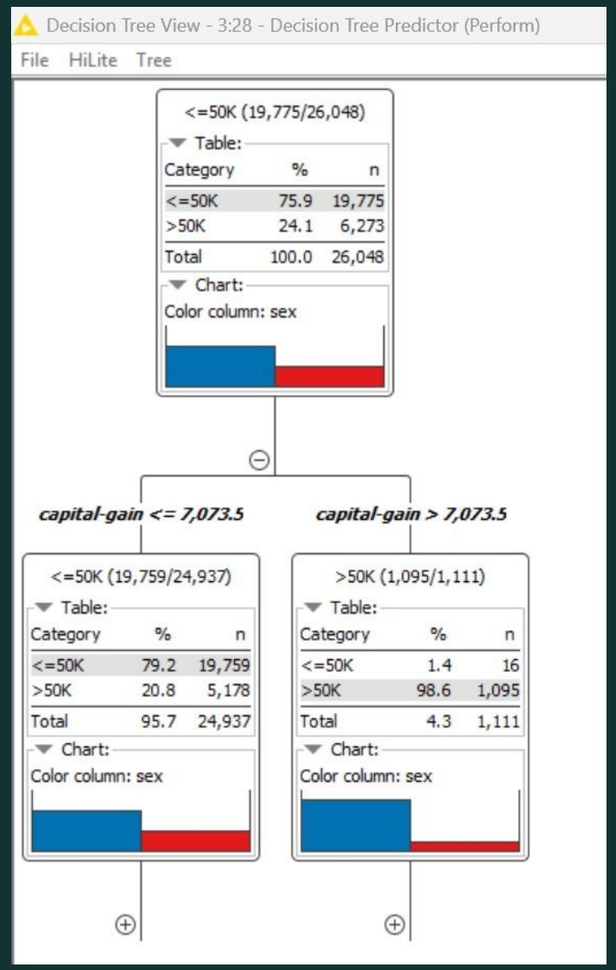
- **Key Insights**:

•Identifies significant predictors of income, such as education level and occupation.
•Visualizes decision paths to understand classification criteria.

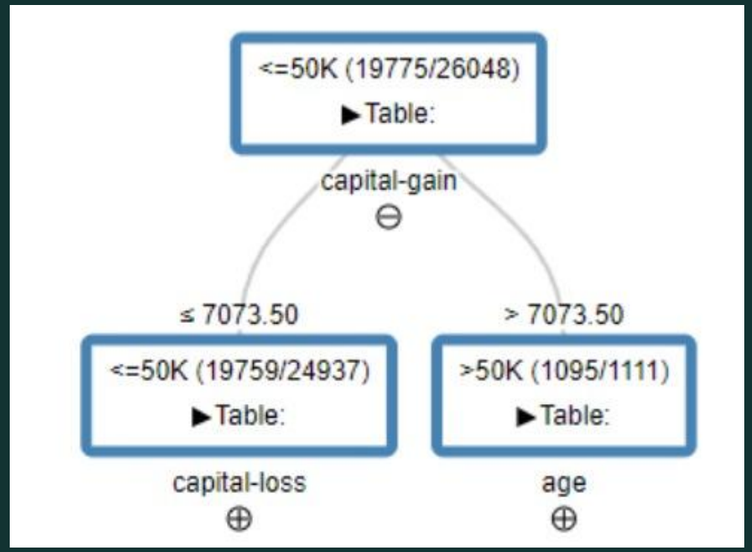
- **Forecasting Capability**:

Uses the trained model to predict income categories for new, unseen data.
•Assesses model accuracy and refines it for improved future predictions.

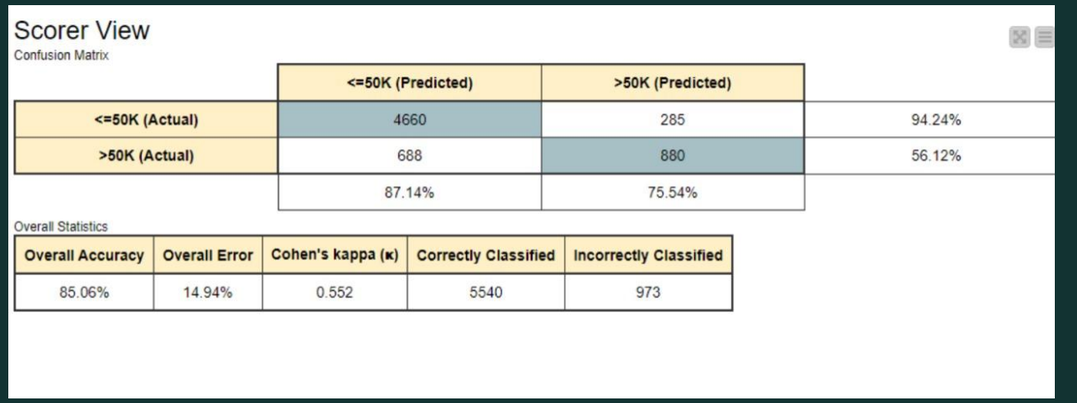
KNIME: Visualizations and Statistics



Decision Tree Learner Output



Decision Tree View in JS



Scorer Confusion Matrix

A "confusion matrix" in machine learning is a table that visually represents how well a classification model performs by showing the breakdown of correct and incorrect predictions, allowing you to see where the model gets "confused" between different classes, and is used to calculate various metrics like accuracy, precision, and recall to evaluate its performance in details.

Conclusion: Results, Insights, and Future Directions

1. Results & Insights

- **Polynomial Regression & Linear Models**: Captured **non-linear trends** effectively, improving prediction accuracy over standard linear regression.
- **EWMA (Exponentially Weighted Moving Average)**: Provided **smoothed trend analysis**, reducing noise and identifying long-term patterns in time-series data.
- The **Decision Tree model** effectively predicts income categories, with **education, occupation, and work hours** as key influencing factors.
- Model accuracy is **satisfactory**, but performance can be improved by **feature engineering** and **hyperparameter tuning**.

2. Future Directions

- **Model Optimization**: Fine-tuning **polynomial degrees, smoothing factors, and regularization techniques** for better accuracy.
- **Hybrid Approaches**: Combining **EWMA with ML models** (like LSTMs or Random Forest) for **enhanced time-series forecasting and anomaly detection**.
- **Enhancing Model Performance**: Incorporate **ensemble learning** (Random Forest, Gradient Boosting) to improve predictions.
- **Expanding Data Scope**: Use **additional datasets** or real-world applications to enhance model generalization and business applicability.