**Phase 5**

|  |  |
| --- | --- |
| **Date** | **31-10-2023** |
| **Team ID** | **3869** |
| **Project Name** | **FUTURE SALES PREDICTION** |

**Table of Contents**

|  |  |
| --- | --- |
| 1 | Introduction |
| 2 | Problem Statement |
| 3 | Design Thinking Approach |
| 4 | Design and Innovation Strategies |
| 4.1 | Data Collection and Feature Engineering |
| 4.2 | Data Pre-Processing |
| 4.3 | Model Selection and Training |
| 4.4 | Visualization Analysis |
| 4.5 | Continuous Training |
| 4.6 | Forecasting Techniques |
| 5 | Program |
| 6 | Conclusion |

**Problem Definition and Design Thinking**

**1.Introduction**

Predicting future sales is a critical task for businesses aiming to optimize operations, allocate resources efficiently, and drive growth. In an ever-evolving market landscape, leveraging data-driven approaches and advanced analytics is essential for accurate sales predictions. This introduction provides an overview of the importance of sales prediction, the role of data and technology, and the potential benefits for businesses**.**

**2.Problem Statement**

Traditional sales forecasting methods struggle to provide precise predictions due to their reliance on historical data and simplistic algorithms. To enhance accuracy and efficiency, there is a need for an advanced predictive modeling framework that integrates diverse data sources, including historical sales data, market trends, consumer behavior, and economic indicators. This framework should yield accurate forecasts while remaining scalable and interpretable, empowering businesses to make data-driven decisions and adapt strategies effectively in a rapidly changing market.

**Key Challenges:**

**1.Dynamic Market Conditions:**

Markets can rapidly change due to economic shifts, technological advancements, or emerging competitors, making it challenging to anticipate sales trends accurately.

**2.Uncertain Factors and Events:**

Unforeseen events such as natural disasters, global health crises, or geopolitical changes can significantly impact consumer behavior and market dynamics, rendering predictions uncertain.

**3.Data Complexity and Variety:**

Managing and analyzing diverse data sources, including structured and unstructured data, consumer behavior, market trends, and economic indicators, present a challenge for creating accurate predictive models.

**4. Seasonal and Cyclical Patterns:**

Sales often follow predictable seasonal or cyclical patterns, but capturing and integrating these patterns effectively into forecasting models can be complex due to variations and anomalies.

**5.Consumer Behavior and Preferences:**

Understanding and predicting shifts in consumer preferences and buying behavior, influenced by societal trends, technological shifts, or marketing strategies, pose a significant challenge.

**6.Competitive Landscape:**

Adapting to changes in the competitive landscape, including new market entrants, shifts in market share, or disruptive business models, is crucial for accurate sales predictions.

**7.Scalability and Data Volume:**

Scaling predictive models to handle large volumes of data efficiently while maintaining accuracy and speed is a critical challenge, especially for growing businesses.

**8.Model Interpretability and Explainability:**

Developing models that are not only accurate but also interpretable and explainable to stakeholders is essential for gaining trust and understanding the predictions made.

**3.Design Thinking Approach**

**Empathize:**

Predicting future sales is akin to navigating a complex maze. Businesses rely on these predictions for survival and growth, influencing crucial decisions, resource management, and overall strategy. The weight of these predictions is immense—impacting employees, stakeholders, and the economy at large. Inaccuracies in predictions can lead to imbalances, affecting livelihoods and stability. Recognizing this burden highlights the critical importance of precise sales forecasting for the business world and the broader societal landscape.

**Actions:**

Enhance Data Quality: Ensure accurate and comprehensive data collection.

1)Apply Advanced Analysis: Use sophisticated analysis techniques to uncover patterns.

2)Implement Machine Learning: Utilize algorithms for robust prediction models.

3)Optimize Features: Identify and use relevant features for modeling.

4)Utilize Ensemble Modeling: Combine predictions from multiple models for better accuracy.

5)Evaluate and Fine-Tune Models: Assess performance and optimize models.

6)Continuous Monitoring and Updating: Keep models updated and relevant.

7)Integrate Feedback Loop: Capture real-time data and user feedback for model refinement.

8)Leverage Collaboration and Knowledge Sharing: Foster a collaborative approach to improve predictions.

9)Invest in Technology: Use advanced tools and technologies for efficient data processing.

10)Encourage Experimentation and Innovation: Stay updated with the latest modeling techniques and innovations for accurate predictions.

**Define:**

Sales Prediction is the use of data analysis and modeling to forecast upcoming sales and revenue based on historical and present trends. It helps businesses anticipate market demand, plan resources, and make informed decisions for optimal growth.

**Objectives:**

Future sales prediction aims to achieve precise forecasting, align resources with demand, optimize operations, enhance performance evaluation, adapt to market trends, and minimize inventory inefficiencies. It guides financial planning, enables personalized marketing, and maximizes the effectiveness of promotions. Additionally, it aids in risk mitigation, successful product launches, and aligning long-term strategies for sustainable growth.

**Ideate:**

with advanced AI algorithms for adaptable sales forecasts. Utilize real-time data analytics to adjust strategies swiftly. Integrate customer insights for precise predictions and tailored marketing.

**Actions:**

To ideate for future sales prediction, first, organize brainstorming workshops involving diverse teams to foster creativity and generate a wide array of innovative ideas. Second, explore cross-industry insights and case studies to extract valuable lessons and adapt successful strategies from different sectors. Third, leverage expert consultations, collaborative workshops, and open innovation platforms to gain diverse perspectives and expertise, sparking creativity and encouraging novel approaches to enhance sales prediction methodologies.

**Prototype**

Create a prototype of the machine learning model and the user interface for price prediction.

**Actions:**

**1. Machine Learning Model Prototype:**

Collect and preprocess historical sales data, train a chosen model (e.g., Linear Regression), and evaluate its performance for initial predictions.

**2. User Interface Prototype:**

Design wireframes, develop an interactive front-end using appropriate technologies, and integrate it with the machine learning model to enable user input and prediction display.

**3. Testing and Iteration:**

Conduct usability tests, gather feedback for UI/UX improvements, and refine both the machine learning model and the interface for enhanced functionality and user satisfaction.

**Test:**

To test future sales prediction models, start by preparing a dataset with historical sales and relevant features. Split the data for training and testing, train the model, and evaluate its accuracy using metrics like MAE and RMSE. Analyze errors, conduct scenario tests, and consider stakeholder feedback to refine the model, ensuring reliable predictions for future sales.

**Actions:**

To rigorously test future sales prediction models, first, segment the historical sales dataset into training and testing sets, dedicating a significant portion for training the prediction model. Next, evaluate the model's accuracy and reliability using established metrics like Mean Absolute Error (MAE) or Mean Squared Error (MSE) by comparing the predicted sales with the actual sales from the testing set. Optionally, conduct comparative testing by comparing multiple prediction models to determine the most efficient and accurate approach for forecasting future sales, aiding in the selection of the optimal model for deployment.

**Implement:**

1. Train a machine learning model using historical sales data and relevant features.

2. Utilize the trained model to forecast future sales based on incoming data and business dynamics.

**Actions:**

1)Train the final machine learning model on the entire dataset.

2)Deploy the model as part of a production-ready web application.

3)Conduct thorough testing to ensure the application is robust and user-friendly.

**Iterate:**

Continuous improvement is essential. Gather user feedback and iterate on the model and interface to enhance accuracy and usability.

**Actions:**

1)Monitor the model's performance and retrain it periodically with updated data.

2)Address user feedback and make necessary improvements to the web interface.

3)Stay informed about advancements in machine learning and real estate pricing models for potential enhancements.

**4.Design and Innovation Strategies**

**4.1. Data Collection and Feature Engineering**

Historical sales data collection is essential; we gather specific data on previous transactions, such as timestamps, product specifications, and sales volume. By using lag features to capture patterns and external events like vacations or economic indicators, feature engineering turns raw data into useful predictions. This phase is essential because it increases the model's accuracy by giving the prediction process useful input variables.

**4.2. Data Pre-processing**

The crucial step of data preparation is where unclean, unstructured data is removed and made ready for analysis. Handling missing values, getting rid of duplicates, and spotting outliers are among the tasks. Additionally, categorical variables are encoded for interoperability with machine learning methods, and numerical features may be scaled. When data is properly preprocessed, it is dependable and ideal for building precise predictive models.

**4.3. Model Selection and Training**

**1. Algorithm Exploration:**

Examine and investigate time-series forecasting techniques that are appropriate given the complexity and specifications of the dataset.

**2. Data Splitting and Preparation:**

Ensure chronological order while dividing the data into training and testing sets.For model compatibility, preprocess features, handle scaling, and encode categorical variables.

**3.Hyperparameter Optimization:**

Using methods like grid search, adjust the model hyperparameters for precision and generality.

**4. Training and Validation:**

Utilize the training set to train the selected model, then use the testing set to evaluate its performance. Use the right criteria to evaluate, and adjust the model as necessary for the best possible forecast of future sales.

**4.4. Visualization Analysis:**

To understand sales trends over time, use exploratory data analysis using visualizations like line charts, histograms, and heatmaps. Investigate subtle trends like seasonality or product-specific behaviours by using interactive tools like Tableau or Matplotlib. Businesses may effectively strategize using visual insights to make informed decisions and comprehend large amounts of data.

**Customer Analysis**

**1. Segmentation:**

To identify different buyer groups, segment clients based on their demographics, region, or shopping habits.

**2. Behavioural Patterns:**

Analyze purchasing patterns, tastes, and product affinities to comprehend consumer behavior and adjust marketing plans accordingly.

**3. Lifetime Value (CLV):**

To determine the profitability of each customer category and to direct resource allocation and targeted marketing initiatives, determine Customer Lifetime Value.

**4. Feedback Integration:**

To learn more about customer satisfaction levels, identify pain points, and improve the entire customer experience, integrate customer feedback and surveys.

**4.5. Continuous Learning**

To succeed in today's changing marketplace, organizations must continuously develop new skills. Companies may efficiently change their strategy to meet changing market demands by keeping up with industry trends and technology developments. By funding employee skill development through workshops and training programs, employers can be guaranteed that their team is competent and up to the task of taking on new challenges. Organizations can also benefit from the insightful feedback provided by both consumers and workers, using this information to improve services, hone goods, and raise overall customer satisfaction. Businesses can take informed decisions and expand sustainably in the face of dynamic business environments by adopting an iterative improvement culture that carefully considers lessons learned from the past.

**4.6. Forecasting Techniques**

Time Series Analysis, which models patterns in sales data over time, Regression Analysis, which takes into account different factors affecting sales, Machine Learning Algorithms like Random Forest and XGBoost for complex, non-linear relationships, Deep Learning Models like Long Short-Term Memory (LSTM) networks for sequential data analysis, and Ensemble Methods, which combine predictions from multiple models for more accurate forecasting, are some forecasting techniques for future sales prediction. The individual characteristics of the sales data and the desired level of accuracy and complexity are important factors for choosing the right technique.

**FLOW DIAGRAM**

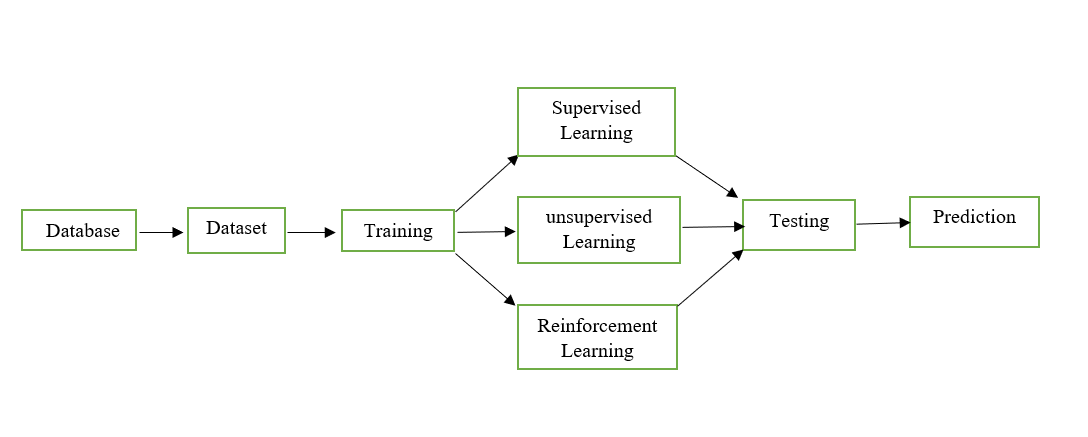


Fig.1 Flow Diagram For Future Sales Prediction

## Importing Dependencies

import numpy as np  
import pandas as pd  
import plotly as px  
import seaborn as sns  
import matplotlib.pyplot as plt  
from sklearn.preprocessing import MinMaxScaler

Import important Python libraries for data science and machine learning:

* numpy for scientific computing
* pandas for data analysis and manipulation
* plotly for interactive data visualization
* seaborn for statistical data visualization
* matplotlib.pyplot for plotting data
* sklearn.preprocessing.MinMaxScaler for scaling numerical features

This code provides the basic foundation for performing a variety of data science and machine learning tasks, such as:

* Loading and cleaning data
* Exploratory data analysis
* Feature engineering
* Model building
* Model evaluation
* Data visualization

# Load Dataset

This code reads the Sales.csv file into a Pandas DataFrame and prints the first five rows.

data = pd.read\_csv('Sales.csv')  
data.head()

TV Radio Newspaper Sales  
0 230.1 37.8 69.2 22.1  
1 44.5 39.3 45.1 10.4  
2 17.2 45.9 69.3 12.0  
3 151.5 41.3 58.5 16.5  
4 180.8 10.8 58.4 17.9

This code returns a tuple of two integers, representing the number of rows and columns in the DataFrame.

data.shape

(200, 4)

The data.info() method in Pandas prints a concise summary of the DataFrame, including the number of rows, columns, data types, memory usage, and range index.

data.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 200 entries, 0 to 199  
Data columns (total 4 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 TV 200 non-null float64  
 1 Radio 200 non-null float64  
 2 Newspaper 200 non-null float64  
 3 Sales 200 non-null float64  
dtypes: float64(4)  
memory usage: 6.4 KB

The data.describe().T code in Pandas transposes the output of the describe() method, which summarizes the central tendency, dispersion, and shape of the numerical values in the DataFrame. Transposing the output means that the columns and rows are swapped.

data.describe().T

count mean std min 25% 50% 75% max  
TV 200.0 147.0425 85.854236 0.7 74.375 149.75 218.825 296.4  
Radio 200.0 23.2640 14.846809 0.0 9.975 22.90 36.525 49.6  
Newspaper 200.0 30.5540 21.778621 0.3 12.750 25.75 45.100 114.0  
Sales 200.0 15.1305 5.283892 1.6 11.000 16.00 19.050 27.0

The data.columns attribute in Pandas returns a list of the column names in the DataFrame.

data.columns

Index(['TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')

# Visualisation of Data

Creates a joint plot with a regression line, showing the relationship between sales and TV advertising. Colors the plot sky blue and the regression line dark blue.

sns.set(style="white")  
sns.jointplot(data=data, x='Sales', y='TV', kind='reg', height=7, color='skyblue', line\_kws={'color':'darkblue'})  
plt.xlabel('Sales', fontsize=14)  
plt.ylabel('TV Advertising', fontsize=14)  
plt.suptitle('Relationship Between Sales and TV Advertising', y=1.02) plt.show()

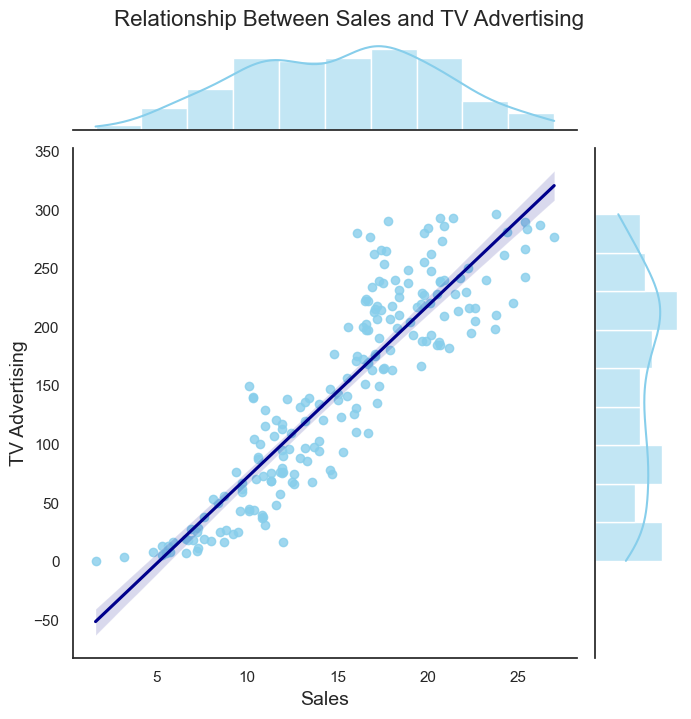


Fig.1 Relationship Between Sales and TV Advertising

This code uses the Seaborn library to create a joint plot with a regression line, showing the relationship between sales and newspaper advertising.

sns.jointplot(data=data, x='Sales', y='Newspaper', kind='reg', height=7, color='skyblue',line\_kws={'color':'darkgreen'})  
plt.xlabel('Sales',fontsize=14)  
plt.ylabel('NewspaperAdvertising',fontsize=14)  
plt.suptitle('Relationship Between Sales and Newspaper Advertising', y=1.02, fontsize=16)  
plt.show()

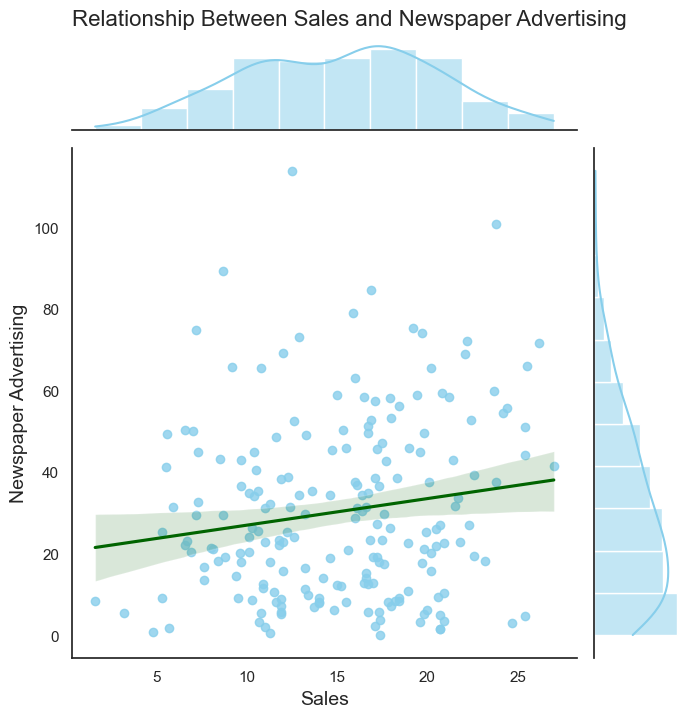


Fig 2. Relationship Between Sales and Newspaper Advertising

Creates a joint plot with a regression line, showing the relationship between sales and radio advertising. Colors the plot sky blue and the regression line dark orange.

sns.set(style="white")  
sns.jointplot(data=data, x='Sales', y='Radio', kind='reg', height=7, color='skyblue', line\_kws={'color':'darkorange'})  
plt.xlabel('Sales', fontsize=14)  
plt.ylabel('Radio Advertising', fontsize=14)  
plt.suptitle('Relationship Between Sales and Radio Advertising', y=1.02, fontsize=16)  
plt.show()

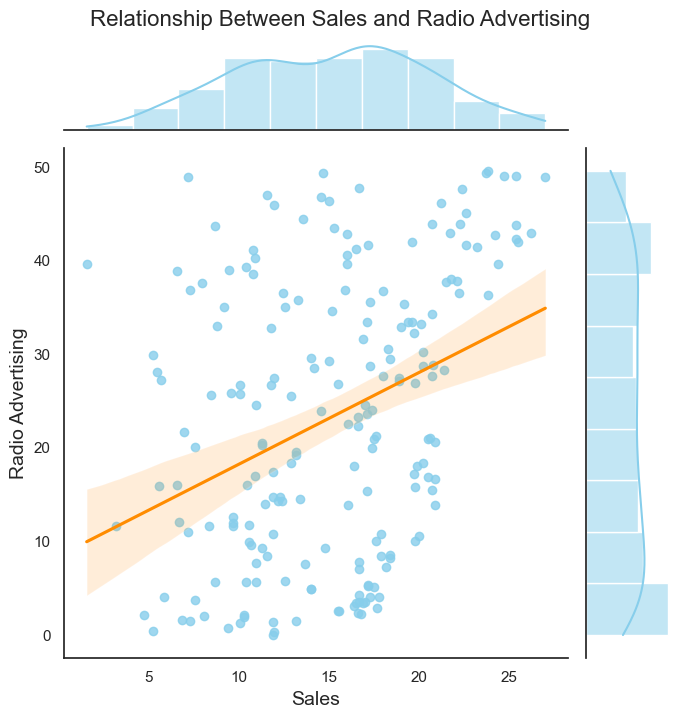


Fig 3. Relationship between Sales and Radio Advertising

Calculates the correlation between all features and sales, sorts by correlation, and creates a styled DataFrame with color gradient to highlight strongest correlations with sales.

correlation = data.corr()  
sales\_correlation = correlation["Sales"].sort\_values(ascending=False)  
styled\_sales\_correlation = sales\_correlation.apply(lambda x: f'{x:.2f}')  
styled\_sales\_correlation = styled\_sales\_correlation.reset\_index()  
styled\_sales\_correlation.columns = ["Feature", "Correlation with Sales"]  
styled\_sales\_correlation.style.background\_gradient(cmap='coolwarm', axis=0)

<pandas.io.formats.style.Styler at 0x1b1017ba250>

The code plt.figure(figsize=(12,8)) creates a new figure with a size of 12 inches by 8 inches. The code sns.pairplot(data) uses the Seaborn library to create a pairplot of the data DataFrame. A pairplot is a visualization that shows the pairwise relationships between all of the variables in a dataset.

plt.figure(figsize=(12,8))  
sns.pairplot(data)

<seaborn.axisgrid.PairGrid at 0x1b17da6b410>

<Figure size 1200x800 with 0 Axes>

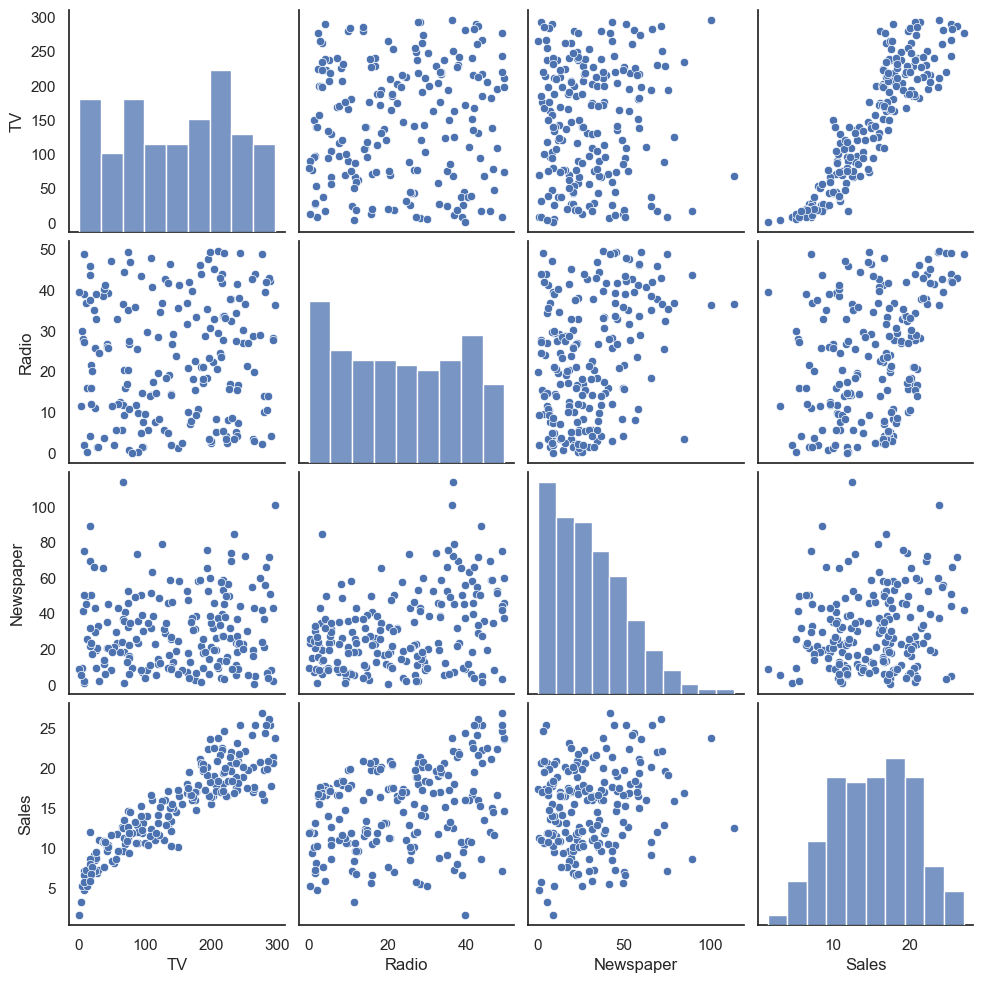


Fig 4. Pair Plot of TV, Radio, Newspaper, and Sales

# Data Preprocessing

To use the hist() method, you simply need to pass in the DataFrame that you want to create a histogram for. The hist() method will automatically create a histogram for each numerical column in the DataFrame. You can also specify the number of bins in the histogram and the color of the bars.

data.hist(figsize=(10,8))

array([[<Axes: title={'center': 'TV'}>,  
 <Axes: title={'center': 'Radio'}>],  
 [<Axes: title={'center': 'Newspaper'}>,  
 <Axes: title={'center': 'Sales'}>]], dtype=object)

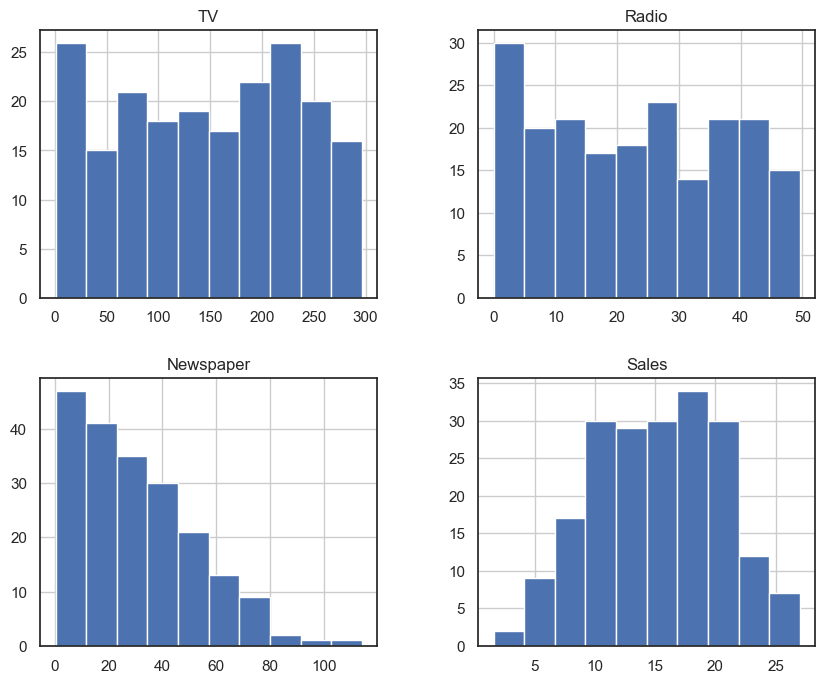


Fig 5. Histograms of TV, Radio, Newspaper, and Sales

# Data normalization

Normalization of numerical features is a common data preprocessing step that is used to scale the values of features to a common range. This can be helpful for machine learning algorithms, as it helps to prevent them from being biased towards features with larger values.

In the given code, the MinMaxScaler() object is used to scale the TV, Radio, and Newspaper columns in the data DataFrame to a range of 0 to 1. This is done by subtracting the minimum value from each column and then dividing by the difference between the maximum and minimum values.

scaler = MinMaxScaler()  
columns\_to\_normalize = ['TV', 'Radio', 'Newspaper']  
data[columns\_to\_normalize] = scaler.fit\_transform(data[columns\_to\_normalize])  
data.head()

TV Radio Newspaper Sales  
0 0.775786 0.762097 0.605981 22.1  
1 0.148123 0.792339 0.394019 10.4  
2 0.055800 0.925403 0.606860 12.0  
3 0.509976 0.832661 0.511873 16.5  
4 0.609063 0.217742 0.510994 17.9

sns.heatmap(data.corr(),annot=True)

<Axes: >

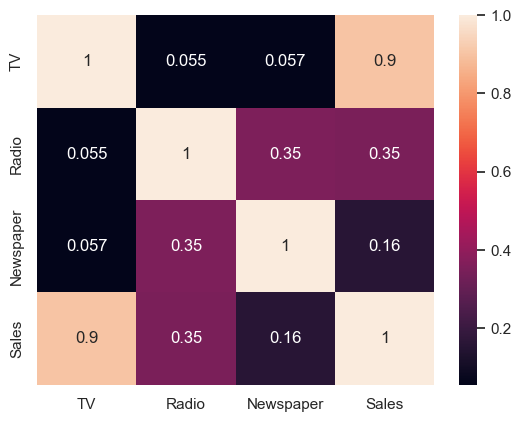


Fig 6. Correlation Heatmap

**5.Program**

# Load Data

import pandas as pd  
  
df = pd.read\_csv(r"C:\Users\thamb\Downloads\archive (1)\Sales.csv")  
df.head()

TV Radio Newspaper Sales  
0 230.1 37.8 69.2 22.1  
1 44.5 39.3 45.1 10.4  
2 17.2 45.9 69.3 12.0  
3 151.5 41.3 58.5 16.5  
4 180.8 10.8 58.4 17.9

Features explanation:

* **TV**: this feature represents the amount of advertising budget spent on television media for a product or service in a certain period, for example in thousands of dollars (USD).
* **Radio**: this feature represents the amount of advertising budget spent on radio media in the same period as TV.
* **Newspaper**: this feature represents the amount of advertising budget spent in newspapers or print media in the same period as TV and Radio.
* **Sales**: This feature represents product or service sales data in the same period as advertising expenditure on TV, Radio and Newspaper.

df.shape

(200, 4)

df.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 200 entries, 0 to 199  
Data columns (total 4 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 TV 200 non-null float64  
 1 Radio 200 non-null float64  
 2 Newspaper 200 non-null float64  
 3 Sales 200 non-null float64  
dtypes: float64(4)  
memory usage: 6.4 KB

df.describe().T

count mean std min 25% 50% 75% max  
TV 200.0 147.0425 85.854236 0.7 74.375 149.75 218.825 296.4  
Radio 200.0 23.2640 14.846809 0.0 9.975 22.90 36.525 49.6  
Newspaper 200.0 30.5540 21.778621 0.3 12.750 25.75 45.100 114.0  
Sales 200.0 15.1305 5.283892 1.6 11.000 16.00 19.050 27.0

# Calculate the correlation  
correlation = df.corr()  
sales\_correlation = correlation["Sales"].sort\_values(ascending=False)  
  
# Format and style the correlation values  
styled\_sales\_correlation = sales\_correlation.apply(lambda x: f'{x:.2f}')  
styled\_sales\_correlation = styled\_sales\_correlation.reset\_index()  
styled\_sales\_correlation.columns = ["Feature", "Correlation with Sales"]  
styled\_sales\_correlation.style.background\_gradient(cmap='coolwarm', axis=0)

<pandas.io.formats.style.Styler at 0x23785318950>

# Data Preprocessing

#### Outlier detection

import seaborn as sns  
import matplotlib.pyplot as plt  
  
# Create the box plot  
plt.figure(figsize=(8, 6))  
sns.boxplot(x='TV', data=df, palette='Blues')  
plt.title('Box Plot of TV Advertising')  
plt.xlabel('TV Advertising Spending')  
plt.grid(axis='x', linestyle='--', alpha=0.6)  
  
# Show the plot  
plt.show()

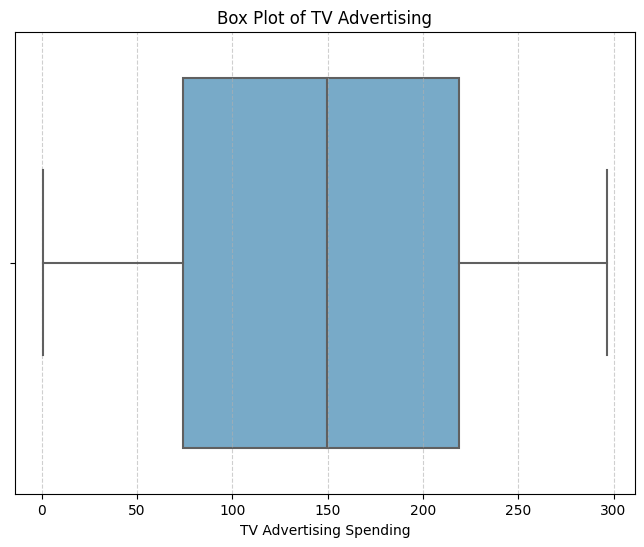


Fig.7 Box Plot of TV Advertising

plt.figure(figsize=(8, 6))  
sns.boxplot(x='Radio', data=df, palette='Oranges')  
plt.title('Box Plot of Radio Advertising')  
plt.xlabel('Radio Advertising Spending')  
plt.grid(axis='x', linestyle='--', alpha=0.6)  
  
# Show the plot  
plt.show()

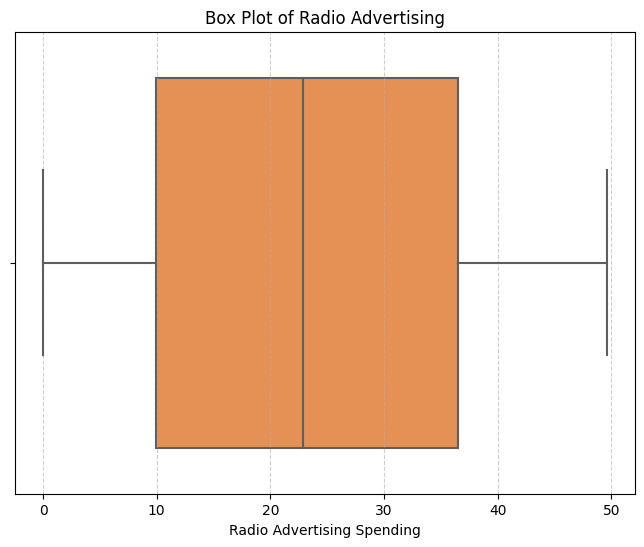


Fig.8 Box Plot of Radio Advertising

# Create the box plot  
plt.figure(figsize=(8, 6))  
sns.boxplot(x='Newspaper', data=df, palette='YlGnBu')  
plt.title('Box Plot of Newspaper Advertising')  
plt.xlabel('Newspaper Advertising Spending')  
plt.grid(axis='x', linestyle='--', alpha=0.6)  
  
# Show the plot  
plt.show()

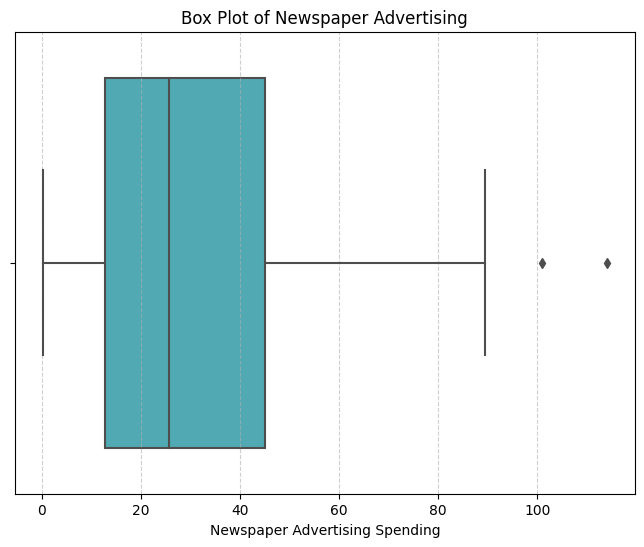


Fig.9 Box Plot of Radio Advertising

There are outliers in the Newspaper feature. To overcome this, we use the Winsorizing technique. Winsorizing is a technique that replaces outlier values with certain predetermined threshold values. We set the threshold value for the Newspaper feature at 2.

import numpy as np  
  
# Ambang batas atas (threshold) untuk Winsorizing  
upper\_threshold = 2 \* np.std(df['Newspaper']) + np.mean(df['Newspaper'])  
  
# Menerapkan Winsorizing pada kolom 'Newspaper'  
df['Newspaper'] = np.where(df['Newspaper'] > upper\_threshold, upper\_threshold, df['Newspaper'])

# Create the box plot  
plt.figure(figsize=(8, 6))  
sns.boxplot(x='Newspaper', data=df, palette='YlGnBu')  
plt.title('Box Plot of Newspaper Advertising')  
plt.xlabel('Newspaper Advertising Spending')  
plt.grid(axis='x', linestyle='--', alpha=0.6)  
  
# Show the plot  
plt.show()

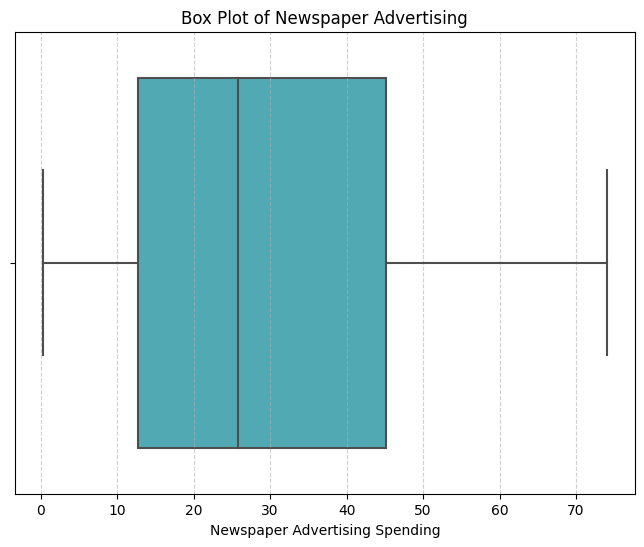


Fig.10 Box Plot of Radio Advertising

#### **Data normalization**

At this stage, we use the min-max technique. Min-Max is a data preprocessing technique used in data analysis and machine learning to convert values in a dataset into a certain range, usually between 0 and 1.

from sklearn.preprocessing import MinMaxScaler  
  
# Create a MinMaxScaler object  
scaler = MinMaxScaler()  
  
# Columns to be normalized (e.g., TV, Radio, Newspaper)  
columns\_to\_normalize = ['TV', 'Radio', 'Newspaper']  
  
# Apply Min-Max normalization to the selected columns  
df[columns\_to\_normalize] = scaler.fit\_transform(df[columns\_to\_normalize])  
df.head()

TV Radio Newspaper Sales  
0 0.775786 0.762097 0.934843 22.1  
1 0.148123 0.792339 0.607851 10.4  
2 0.055800 0.925403 0.936200 12.0  
3 0.509976 0.832661 0.789664 16.5  
4 0.609063 0.217742 0.788307 17.9

# Modelling and Evaluation

At the modeling stage, we use 5 algorithms for comparison, namely Linear Regression, Ridge Regression, Lasso Regression, Decision Tree, and Random Forest and for evaluation using MSE, RMSE, MAE and R-Squared.

X = df[['TV', 'Radio', 'Newspaper']]  
y = df['Sales']

from sklearn.model\_selection import cross\_val\_score  
  
num\_folds = 5  
def perform\_cross\_validation(model, X, y, num\_folds):  
 mse\_scores = -cross\_val\_score(model, X, y, cv=num\_folds, scoring='neg\_mean\_squared\_error')  
 rmse\_scores = np.sqrt(mse\_scores)  
 mae\_scores = -cross\_val\_score(model, X, y, cv=num\_folds, scoring='neg\_mean\_absolute\_error')  
 r2\_scores = cross\_val\_score(model, X, y, cv=num\_folds, scoring='r2')  
   
 return mse\_scores, rmse\_scores, mae\_scores, r2\_scores

# Linear Regression

In machine learning, a method called linear regression is used to predict a continuous outcome variable, also known as a dependent variable, from one or more predictor factors, also known as independent variables. It makes the assumption that the inputs and the output have a linear relationship. The best-fit line that minimizes the variation between the expected and actual values is the one to be found. For jobs like price forecasting, trend analysis, and sales forecasting, this method is frequently employed.

from sklearn.linear\_model import LinearRegression, Ridge, Lasso  
linear\_model = LinearRegression()  
linear\_mse, linear\_rmse, linear\_mae, linear\_r2 = perform\_cross\_validation(linear\_model, X, y, num\_folds)  
print("Linear Regression:")  
print(f"Average MSE: {np.mean(linear\_mse) / np.mean(y) \* 100:.2f}%")  
print(f"Average RMSE: {np.mean(linear\_rmse) / np.mean(y) \* 100:.2f}%")  
print(f"Average MAE: {np.mean(linear\_mae) / np.mean(y) \* 100:.2f}%")  
print(f"Average R-squared: {np.mean(linear\_r2) \* 100:.2f}%")  
print("\n")

Linear Regression:  
Average MSE: 18.90%  
Average RMSE: 11.01%  
Average MAE: 8.38%  
Average R-squared: 89.53%

# Ridge Regression

In machine learning, ridge regression is a regularization approach that keeps linear regression models from overfitting. In order to discourage large coefficients for the input features, it adds a penalty term (L2 regularization) to the linear regression cost function. This facilitates the handling of multicollinearity and yields a more accurate and stable model, particularly in the case of high dimensionality datasets. Ridge regression is a crucial component in enhancing the generalization and performance of models on intricate datasets.

ridge\_model = Ridge(alpha=1.0)  
ridge\_mse, ridge\_rmse, ridge\_mae, ridge\_r2 = perform\_cross\_validation(ridge\_model, X, y, num\_folds)  
print("Ridge Regression:")  
print(f"Average MSE: {np.mean(ridge\_mse) / np.mean(y) \* 100:.2f}%")  
print(f"Average RMSE: {np.mean(ridge\_rmse) / np.mean(y) \* 100:.2f}%")  
print(f"Average MAE: {np.mean(ridge\_mae) / np.mean(y) \* 100:.2f}%")  
print(f"Average R-squared: {np.mean(ridge\_r2) \* 100:.2f}%")  
print("\n")

Ridge Regression:  
Average MSE: 19.67%  
Average RMSE: 11.20%  
Average MAE: 8.54%  
Average R-squared: 89.19%  
  
**Lasso Regression**

A machine learning method called lasso regression is employed for regularization as well as variable selection. It encourages sparsity in the coefficient values by adding a penalty term (L1 regularization) to the linear regression cost function, much like ridge regression. Lasso works to reduce overfitting by setting the coefficients of less significant features to zero, which is especially helpful in high-dimensional datasets. It is useful for selecting features and creating more straightforward, understandable models when there are a lot of input variables.

lasso\_model = Lasso(alpha=1.0)   
lasso\_mse, lasso\_rmse, lasso\_mae, lasso\_r2 = perform\_cross\_validation(lasso\_model, X, y, num\_folds)  
print("Lasso Regression:")  
print(f"Average MSE: {np.mean(lasso\_mse) / np.mean(y) \* 100:.2f}%")  
print(f"Average RMSE: {np.mean(lasso\_rmse) / np.mean(y) \* 100:.2f}%")  
print(f"Average MAE: {np.mean(lasso\_mae) / np.mean(y) \* 100:.2f}%")  
print(f"Average R-squared: {np.mean(lasso\_r2) \* 100:.2f}%")  
print("\n")

Lasso Regression:  
Average MSE: 115.55%  
Average RMSE: 27.51%  
Average MAE: 22.39%  
Average R-squared: 35.98%

# Decision Tree

In machine learning, a decision tree is a predictive model that associates features with results. It divides the data recursively according to features, forming a structure like a tree, with each internal node denoting a choice made in response to a feature and each leaf node denoting the expected result. Decision trees are useful tools in machine learning for decision-making processes because they are adaptable, comprehensible, and frequently utilized for classification and regression problems.

from sklearn.tree import DecisionTreeRegressor  
tree\_model = DecisionTreeRegressor(max\_depth=None, random\_state=0)   
tree\_mse, tree\_rmse, tree\_mae, tree\_r2 = perform\_cross\_validation(tree\_model, X, y, num\_folds)  
print("Decision Trees:")  
print(f"Average MSE: {np.mean(tree\_mse) / np.mean(y) \* 100:.2f}%")  
print(f"Average RMSE: {np.mean(tree\_rmse) / np.mean(y) \* 100:.2f}%")  
print(f"Average MAE: {np.mean(tree\_mae) / np.mean(y) \* 100:.2f}%")  
print(f"Average R-squared: {np.mean(tree\_r2) \* 100:.2f}%")  
print("\n")

Decision Trees:  
Average MSE: 16.73%  
Average RMSE: 10.40%  
Average MAE: 7.56%  
Average R-squared: 90.65%  
  
**Random Forest**

A machine learning ensemble learning technique called Random Forest uses several decision trees. By training each tree on a different subset of the data and features, it creates a forest of trees. When making predictions, it either uses a majority vote (for classification) or averages the outcomes from individual trees (for regression). For ML applications like feature selection, regression, and classification, Random Forests are frequently utilized due to their robustness and ability to manage intricate relationships in data.

from sklearn.ensemble import RandomForestRegressor  
forest\_model = RandomForestRegressor(n\_estimators=100, random\_state=0)   
forest\_mse, forest\_rmse, forest\_mae, forest\_r2 = perform\_cross\_validation(forest\_model, X, y, num\_folds)  
print("Random Forest:")  
print(f"Average MSE: {np.mean(forest\_mse) / np.mean(y) \* 100:.2f}%")  
print(f"Average RMSE: {np.mean(forest\_rmse) / np.mean(y) \* 100:.2f}%")  
print(f"Average MAE: {np.mean(forest\_mae) / np.mean(y) \* 100:.2f}%")  
print(f"Average R-squared: {np.mean(forest\_r2) \* 100:.2f}%")

Random Forest:  
Average MSE: 10.32%  
Average RMSE: 8.09%  
Average MAE: 5.99%  
Average R-squared: 94.27%

**6.Conclusion:**

Future sales prediction relies on a complex method. To comprehend previous trends and the dynamics of the market today, historical data analysis and market research are the first steps. Accurate forecasting requires a variety of data sources, including customer feedback and sales reports. Time series analysis and feature selection, respectively, highlight important variables and temporal patterns. Metrics like MAE and RMSE are used to evaluate how well machine learning models and cross-validation produce accurate predictions. A complete picture is provided by scenario analysis and the identification of important sales-influencing variables. In order to continuously refine the model and ensure its adaptability to changes in the market, feedback loops and continuous monitoring are essential. To sum up, a strong sales prediction strategy combines sophisticated modelling , market insights, and data analysis to enable precise and flexible forecasting.

**References**

[1] J. Han, Y. Liu, and X. Sun, A scalable random forest algorithm based on Map Reduce, 2013

IEEE 4th Int. Conf. Software Engin. Serv. Sci. Beijing (2013) 849–852.

[2] Bohdan M. Pavlyshenko (2018),” Machine Learning models for sales time series forecasting”

[3] Sekban, Judi. "Applying machine learning algorithms in sales prediction." (2019).

[4] Giering, Michael. "Retail sales prediction and item recommendations using customer demographics at store level." *ACM SIGKDD Explorations Newsletter* 10, no. 2 (2008): 84-89.