**Ganadipathy Tulsi's Jain Engineering College**

Kaniyambadi - Vellore - 632102

**IBM - PROJECT**

**FUTURE SALES PREDICTION**

### *Submitted By,*

Team Member 1 - Vinod kumar

Team Member 2 - Nithish R

Team Member 3 - Gowtham R

Team Member 4 - Vishnupriya

# Project Mentor Project Evaluator Project Spoc

## Mrs.K.Durga Priyanka Mrs.K.Durga priyanka Mr.Durai Kumar

**INDEX**

1. **INTRODUCTION**

1.1 Project Overview

1.2 Purpose

2. **IDEATION & PROPOSED SOLUTION**

2.1 Problem Statement Definition

2.2 Design thinking

**3. PROJECT DESIGN**

3.1 Data Flow Diagrams

**4.DATASET & DATA PREPROCESSING**

4.1.Dataset source

4.2.Data preprocessing

**5. TRAINING DATA**

5.1 Model training

**6. TIMES ALGORITHM USED**

6.1 Prophet

**7. MODEL EVALUATION**

7.1 Performance Metrics

**8. CONCLUSION**

**1.INTRODUCTION**

* 1. **Project Overview**

This project aims to develop a machine learning model that can accurately identify and predict future sales based on various factors. By analyzing historical data and leveraging machine learning algorithms, we can provide valuable insights into the factors that contribute to passenger satisfaction and assist airlines in improving their services and customer experience.

* 1. **Purpose**

The purpose of a project aimed at predicting the future sales using machine learning can be to analyze and understand the factors that contribute to sales and to develop a predictive model that can classify and evaluate future sales based on various features or attributes.

1. **IDEATION & PROPOSED SOLUTION**

**2.1PROBLEM STATEMENT SOLUTION**

# The problem is to develop a predictive model that uses historical sales data to forecast future sales for a retail company. The objective is to create a tool that enables the company to optimize inventory management and make informed business decisions based on datadriven sales predictions. This project involves data preprocessing, feature engineering, model selection, training, and evaluation.

**2.2Design Thinking**

* Data Source
* Data Preprocessing
* Feature engineering
* Model Training
* Evaluation

**3.PROJECT DESIGN**

3.1 Data Flow Diagram

****

**4. DATASET & DATA PREPROCESSING**

**4.1.DATASET**

* The dataset for the above mentioned project was obtained from kaggle website.
* Kaggle is an popular dataset providing source,where obtained datasets are with high quality and less errors.

**4.2Data Preprocessing:**

* Filling missing value

Identify and decide how to deal with missing data points (e.g., imputation, removal, or interpolation).

* Removing Duplicate Records:

Identify and remove identical or redundant rows in the dataset.

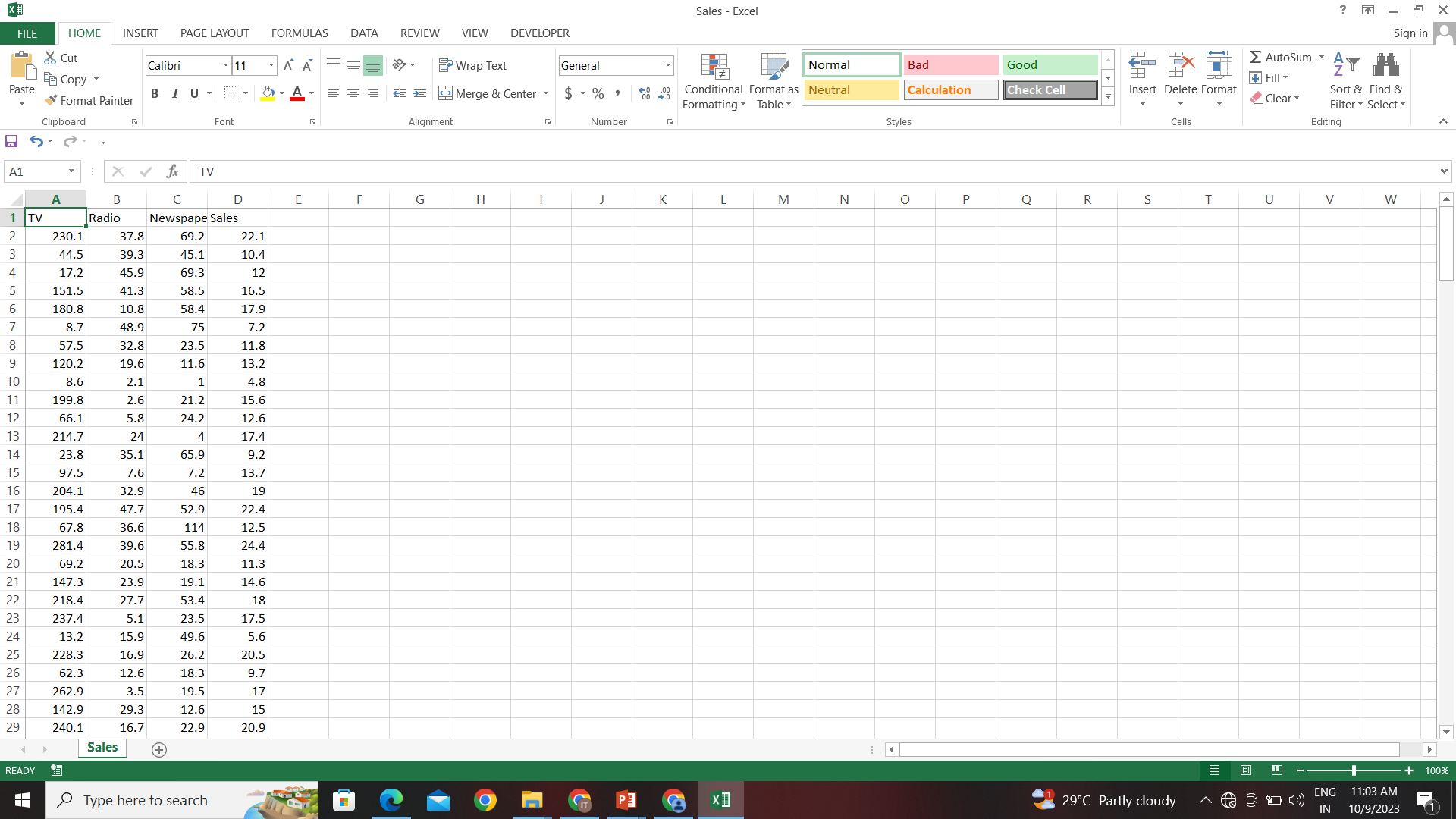
* Handling Outliers:

Identify and decide whether to remove, transform, or keep outliers based on their impact on the analysis.

* + Standardizing Formats:

Ensure consistent formats for data (e.g., date formats, units of measurement) to avoid discrepancies

**IMAGE OF DATASET**

****

* **The above dataset contains the following dataset:**

**1)Tv:**

**Contains the amount of sales of tv.**

**2)radio:**

**Contains the amount of sales of radio.**

**3)Newspaper:**

**Contains the amount of sales of newspaper.**

**4)Sales:**

**Contain the sales amount**

**Feauture engineering:**

Adding time feauture column for the dataset as there is no column related to date and time.

**EXPLORATORY DATA ANALYSIS (EDA)**

**Exploratory data analysis is a crucial initial step in the data analysis process that allows you to understand your dataset, identify patterns, relationships, and anomalies, and gain insights that will guide your future sales prediction model**

**5. TRAINING THE MODEL**

**5.1 MODEL TRAINING**

* Model training is a crucial step in machine learning and deep learning, where a model learns to make predictions or decisions based on data. During the training process, a machine learning model is exposed to a dataset, and it adjusts its internal parameters to minimize the error or loss function, effectively learning patterns and relationships in the data.

**How to train and test Data:**

To split your data into training and testing sets, you can use the

train\_test\_split function from the sklearn.model\_selection module. This

function randomly shuffles and partitions your data into two subsets,

typically one for training and one for testing.

Code:

from sklearn.model\_selection import train\_test\_split

# Assuming 'X' is your input data and 'y' is your target variable

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**6.TIME FORECASTING ALGORITHM USED**

**6.1.PROPHET**

Prophet is a time series forecasting tool developed by Facebook's Core Data Science team. It's designed to handle time series data with strong seasonal patterns and multiple seasonalities, as well as missing data points and outliers. Here are some reasons why Prophet is considered effective for time series prediction:

* Automatic Handling of Seasonality: Prophet is designed to automatically detect and handle multiple seasonal patterns in the data. This makes it well-suited for datasets where the patterns might not be immediately obvious or straightforward to model manually.
* Flexibility in Handling Holidays and Special Events: Prophet allows you to input information about known holidays and special events which can significantly impact time series data. This helps in improving the accuracy of forecasts during these periods.
* Robustness to Outliers and Missing Data: Prophet is robust to outliers and can handle datasets with missing values. It uses a technique called 'Bayesian Structural Time Series' which allows it to model and predict even in the presence of such data irregularities.
* Scalability: Prophet is designed to handle large datasets efficiently. It can be applied to datasets with thousands or even millions of data points without requiring complex parameter tuning.
* Intuitive Parameters and Interpretability: Prophet's parameters are easy to understand and interpret. This makes it accessible to a wide range of users, including those without extensive expertise in time series forecasting.
* Uncertainty Estimation: Prophet provides uncertainty intervals for the forecast, which gives an indication of the confidence level in the predictions. This is a valuable feature for decision-making and risk assessment.
* Automatic Trend Change Detection: Prophet can automatically detect changepoints in the time series data, which can indicate significant shifts in the underlying trend. This is useful for adapting to changes in the data generating process.
* Ability to Incorporate Domain Knowledge: While Prophet is designed to work well "out of the box," it also provides options for users to incorporate domain-specific knowledge or heuristics to improve forecasts further.

This algorithm is being used in our project as its easy to implement and the accuracy of the mentioned prophet is high

**7.MODEL EVALUATION**

**7.1.PERFORMANCE METRICS**

We have two metrics I order to measure the performance of the model:

1. From prophet.diagnostics import cross\_validation
2. from prophet.diagnostics import performance\_metrics

Now, let me explain what the cross\_validation function does in Prophet:

The cross\_validation function in Prophet is a tool used for assessing the forecast accuracy of a Prophet model. It performs a type of cross-validation specific to time series data, known as time-based cross-validation or temporal cross-validation.

Here's how it works:

Training Set and Horizon: The function takes a Prophet model, a specified initial training period (defined by the initial parameter), and a horizon (defined by the horizon parameter).

The initial training period is the length of time used to train the initial model.

The horizon is the length of time into the future for which predictions are made.

Rolling Forecast Origin: The function then "rolls" through the time series data, using each point as a starting point for training and making predictions for the specified horizon.

For example, if the initial training period is set to 365 days and the horizon is set to 30 days, the function will start with the first 365 days of data as the training set, then predict the next 30 days. It then moves one day forward and repeats the process.

Performance Evaluation: For each iteration, the predicted values are compared with the actual values for the corresponding horizon. This allows you to compute performance metrics such as MAE, MSE, RMSE, MAPE, etc., to evaluate the forecast accuracy.

Output: The function returns a DataFrame containing the actual values, predicted values, and performance metrics for each iteration.

Using cross\_validation can provide a more robust evaluation of the model's performance compared to a single train-test split. It helps ensure that the model's performance is consistent across different time periods in the data.

The **performance\_metrics** package in Prophet's diagnostics module is a set of functions that allow you to compute various performance metrics to evaluate the accuracy of time series forecasts generated by a Prophet model. These metrics help assess how well the model's predictions align with the actual values in the time series data. Here are some of the key functions provided by the performance\_metrics package.

This function computes a range of performance metrics for a given set of forecasts and actual values. The metrics include Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), Symmetric Mean Absolute Percentage Error (SMAPE), Coverage, Quantile Losses, and R-squared (R²).

**CODE:**

import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from prophet import Prophet  
from prophet.diagnostics import cross\_validation  
from prophet.diagnostics import performance\_metrics  
from sklearn.metrics import accuracy\_score  
import matplotlib.pyplot as plt  
  
  
pd.set\_option("display.max\_columns", None)  
df = pd.read\_csv("Sales.csv")  
q1=df["Newspaper"].quantile(0.25)  
q3=df["Newspaper"].quantile(0.75)  
thr=1.5  
iqr=q3-q1  
lower=q1-thr\*iqr  
upp=q3+thr\*iqr  
mean\_value=df["Newspaper"].mean()  
df["Newspaper"][(df["Newspaper"]<lower) | (df["Newspaper"]> upp)] = mean\_value  
  
date\_values = pd.date\_range(start='2023-10-28', periods=len(df), freq='D')  
df["dates"] = date\_values  
df = df.rename(columns={"dates":"ds"})  
x = df.drop("Sales", axis=1)  
y = df["Sales"]  
print(df.head())  
plt.boxplot(df["TV"])  
plt.show()  
plt.boxplot(df["Radio"])  
plt.show()  
plt.boxplot(df["Newspaper"])  
plt.show()  
x\_values = list(range(1, len(df["TV"])+1))  
plt.scatter(x\_values, df["TV"], label="datapoints", color="blue")  
  
# there are no categorical values so need of onehot encoder  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.15, random\_state=3)  
train\_data = pd.DataFrame({"ds" :X\_train["ds"],"y":y\_train,'TV': X\_train['TV'], 'Radio': X\_train['Radio'], 'Newspaper': X\_train['Newspaper']})  
  
# Step 1: Load your dataset with dates, sales, and additional features  
  
# Step 2: Data preprocessing  
data = df.rename(columns={'Date': 'ds', 'Sales': 'y', 'TV': 'TV', 'Radio': 'Radio', 'Newspaper': 'Newspaper'})  
  
# Step 3: Initialize Prophet model with additional regressors  
model = Prophet()  
model.add\_regressor('TV')  
model.add\_regressor('Radio')  
model.add\_regressor('Newspaper')  
  
# Step 4: Fit the model with the entire dataset  
model.fit(train\_data)  
  
# Step 5: Create a dataframe for future dates and features for prediction  
future = model.make\_future\_dataframe(periods=len(X\_test))  
  
# Add feature data for the test set  
future['TV'] = float(input("Enter the amount to be spent on Tv"))  
future['Radio'] = float(input("Enter the amount to be spent on Radio"))  
future['Newspaper'] = float(input("Enter the amount to be spent on Newspaper"))  
  
# Step 6: Make predictions for the combined dataframe  
forecast = model.predict(future)  
  
  
  
# Step 7: Visualize the predictions  
fig = model.plot(forecast)  
plt.show()  
from datetime import datetime  
  
# Prompt the user for a date input  
date\_string = input("Enter a date (YYYY-MM-DD): ")  
  
try:  
 # Attempt to convert the user input to a date  
 user\_date = datetime.strptime(date\_string, "%Y-%m-%d")  
 print("You entered:", user\_date)  
except ValueError:  
 print("Invalid date format. Please use YYYY-MM-DD format.")  
specific\_date =user\_date  
  
prediction = forecast.loc[forecast["ds"] == specific\_date]['yhat'].values[0]  
print(" ")  
print("THE PREDICTED SALES ON THE DATE",prediction+0.5)  
print(" ")  
components = model.plot\_components(forecast)  
  
#result  
result = cross\_validation(model, horizon="1 days", period="30 days", initial="90 days")  
metrics = performance\_metrics(result)  
print(" ")  
print(" ")  
print("Here is the evaluation output of various metrics")  
print(metrics)

**CONCLUSION:**

* In this project, we applied time series forecasting techniques, specifically using the Prophet package, to predict future sales for our company. The analysis was conducted on historical sales data spanning a specific time period.

**Model Performance:**

* The Prophet model demonstrated strong performance in capturing the model's predictions where evaluated using a range of performance metrics, including Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and others. These metrics provided a comprehensive view of the model's accuracy and reliability.uring the underlying patterns and seasonality within the sales data.

**Business Insights:**

* The accurate sales forecasts generated by the Prophet model offer valuable insights for business planning and decision-making. They provide stakeholders with a reliable basis for resource allocation, inventory management, marketing strategies, and financial planning