MICRO PROJECT

DATA ANALYTICS LAB(CDT305)

MARKET BASKET ANALYSIS USING APRIORI ALGORITHM AND FREQUENT PATTERN GROWTH

Submitted by

PAUL BINU (MAC22CD048)

PAULU WILSON (MAC22CD049)

RONAL GEORGE SHOEY (MAC22CD051)

To

The APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY IN
COMPUTER SCIENCE AND ENGINEERING
(DATA SCIENCE)

KNOWLEDGE IS POWER

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

MAR ATHANASIUS COLLEGE OF ENGINEERING

KOTHAMANGALAM , KERALA-686 666 JANUARY 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

MAR ATHANASIUS COLLEGE OF ENGINEERING

KOTHAMANGALAM , KERALA-686 666

CERTIFICATE

This is to certify that the report entitled "Market Basket Analysis with Apriori Algorithm and Frequent Pattern Growth (Fp-Growth) on Outdoor Product Sales Data**"** submitted **by Mr.Paul Binu(**MAC22CD048**), Mr.Paulu Wilson(**MAC22CD049**) & Mr.Ronal George Shoey (**MAC22CD051**)** to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science & Engineering for the academic year 2023-2024 is a bonafide record of the micro project presented by them under our supervision and guidance. This report in any form has not been submitted to any other university or institute for any purpose.

………………………… ………..…………………………

Prof. Richu Shibu **Prof. Joby George**

Staff in charge **Head of the Department**

ACKNOWLEDGEMENT

First and foremost, we sincerely thank God Almighty for his grace for the successful and timely completion of the micro project. We express our sincere gratitude and thanks to our Principal Dr. Bos Mathew Jos and Head of the Department Prof. Joby George for providing the necessary facilities and their encouragement and support. We owe special thanks to the faculty in charge Prof. Richu Shibu for their corrections, suggestions and sincere efforts to coordinate the micro project under a tight schedule. We express our sincere thanks to staff members in the Department of Computer Science and Engineering who have taken sincere efforts in helping me to conduct this seminar. Finally, we would like to acknowledge the heartfelt efforts, comments, criticisms, co-operation and tremendous support given to me by our dear friends during the preparation of the seminar and also during the presentation without whose support this work would have been all the more difficult to accomplish.

ABSTRACT

In today's competitive business environment, marketing campaigns play a vital role in attracting and retaining customers. These campaigns can be broadly categorized into traditional and online marketing strategies, each with distinct strengths. This report presents the development of a **marketing campaign prediction system** that leverages machine learning algorithms to predict the outcomes of both traditional and online campaigns.

We implemented four machine learning models—**Decision Tree, Random Forest, Linear Regression,** and **K-Nearest Neighbors (KNN)**—and evaluated their performance based on **Root Mean Square Error (RMSE)**. The models were trained on historical campaign data and tested on unseen data to determine the best model for each campaign type. Results indicate that **K-Nearest Neighbors (KNN)** is the most effective model, delivering the lowest RMSE for both traditional and online campaign predictions.

By providing accurate predictions, the system enables businesses to make informed decisions, optimize campaign strategies, and allocate resources efficiently, thereby improving campaign success. This work demonstrates the power of data-driven approaches in enhancing marketing efforts, combining the strengths of both traditional and digital strategies.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Marketing campaigns are essential for businesses to drive customer engagement, sales, and overall brand awareness. These-campaigns can be categorized into two broad types:

- **Traditional Marketing Campaigns**: These involve offline marketing methods such as print advertisements, TV and radio spots, and direct mail. These methods are effective for reaching audiences who may not be as engaged in digital spaces, such as local communities or older demographics.
- **Online Marketing Campaigns**: These involve digital methods like social media advertising, email marketing, search engine optimization (SEO), and content marketing. Digital campaigns offer better targeting, tracking, and real-time optimization, making them a powerful tool in the modern marketing landscape.

In this project, we developed a **marketing campaign prediction system** to analyze the performance of both traditional and online marketing campaigns using machine learning models. The system predicts campaign outcomes based on different input features, providing businesses with valuable insights to enhance their marketing strategies.

CHAPTER 2

SYSTEM DESIGN

2.1 System Architecture

The marketing campaign prediction system is divided into two main subsystems: one for traditional marketing and one for online marketing. The system processes campaign data, applies machine learning models, and outputs predictions of campaign performance based on metrics such as **sales**, **engagements**, and **customer interactions**.

2.2 Key components of the system architecture include:

- **Data Input**: Campaign data from various sources (e.g., sales figures, customer demographics, online interactions) are fed into the system.
- **Preprocessing**: The data is cleaned, normalized, and prepared for analysis.
- **Model Training**: Machine learning models (Decision Tree, Random Forest, Linear Regression, and KNN) are trained using historical data from past campaigns.
- **Prediction**: Once trained, the models predict the performance of new or ongoing campaigns.
- **Evaluation**: Model performance is evaluated using metrics like **RMSE** (Root Mean Square Error), which measures how well the model fits the data.

2.3 Components Overview

- **Data Preprocessing**: This stage involves handling missing values, normalizing data, and transforming features into a format suitable for machine learning models.
- **Model Training**: We used historical campaign data to train machine learning models. This includes splitting the data into training and testing sets and fitting the models to learn from the data.
- **Prediction and Evaluation**: After training, the models predict the performance of upcoming or ongoing campaigns. Their accuracy is evaluated using **RMSE** to ensure reliability.

2.4 ALGORITHMS

To predict campaign performance, we used four machine learning algorithms. Each model has its strengths, and we evaluated them based on how well they predicted traditional and online marketing campaign outcomes.

2.4.1 Decision Tree

Decision Trees are intuitive and straightforward machine learning models. They split the dataset into smaller and smaller subsets based on decision rules derived from the features. At each node of the tree, the algorithm selects the feature that best separates the data into meaningful segments.

- **Strengths**: Easy to interpret and implement, handles categorical and numerical data well.
- **Weaknesses**: Can overfit the data if not pruned.
- **Traditional Campaign RMSE**: 0.3999999999999986
- **Online Campaign RMSE**: 0.0008333333333333318

2.4.2 Random Forest

Random Forest is an ensemble method that improves upon Decision Trees by building multiple trees and aggregating their predictions. By averaging the results of multiple trees, Random Forest reduces the risk of overfitting and generally provides better accuracy than a single decision tree.

- **Strengths**: Reduces overfitting, handles large datasets and high-dimensional spaces well.
- **Weaknesses**: Computationally expensive compared to a single decision tree.
- **Traditional Campaign RMSE**: 0.9570000000000132
- **Online Campaign RMSE**: 0.00020277777777774966

2.4.3 Linear Regression

Linear Regression is a simple yet powerful algorithm that models the relationship between one or more input features (independent variables) and the target variable (dependent variable) by fitting a straight line to the data. It is particularly useful for predicting numerical outcomes.

- **Strengths**: Simple, interpretable, and computationally efficient.
- **Weaknesses**: Limited to linear relationships; may not perform well with complex data.
- **Traditional Campaign RMSE**: 0.6253588516746369
- **Online Campaign RMSE**: 0.0007986106651210328

2.4.4 K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a non-parametric algorithm that makes predictions by finding the most similar data points (neighbors) and averaging their outcomes. KNN is highly effective in cases where the decision boundaries are non-linear.

- **Strengths**: Simple, effective for both classification and regression, no need for training.
- **Weaknesses**: Can be computationally expensive for large datasets.
- **Traditional Campaign RMSE**: 0.33333333333333215
- **Online Campaign RMSE**: 9.259259259259203e-05

2.5 Tools and Libraries

The system was developed using a variety of tools and libraries to support data manipulation, machine learning, and visualization.

- **Python**: The core programming language used for implementation.
- **Scikit-learn**: A machine learning library that provides tools for building and evaluating models like Decision Tree, Random Forest, Linear Regression, and KNN.
- **Pandas**: A library for data manipulation and analysis, allowing easy handling of large datasets.
- **NumPy**: Used for numerical computations.
- **Matplotlib and Seaborn**: Libraries used for data visualization, enabling us to plot results and observe trends.

CHAPTER 3

PROGRAM AND RESULT

3.1.CODE IMPLEMENTATION

Model

```
# Required Libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.linear_model import LinearRegression

from sklearn.neighbors import KNeighborsRegressor

from sklearn.metrics import mean_squared_error, r2_score

# Load datasets from CSV files

traditional_df = pd.read_csv('traditional_campaign.csv')  # Load traditional campaign data

online_df = pd.read_csv('facebook_campaign.csv')  # Load online campaign data

# Data Cleaning and Preprocessing (Traditional Campaign Dataset)

traditional_df['MarketSize'] = traditional_df['MarketSize'].map({'Small': 1, 'Medium': 2, 'Large':
3})

# Add Click-Through Rate (CTR) to Online Dataset
```

```
online_df['ctr'] = online_df['clicks'] / online_df['impressions']

# Features and Target for Traditional and Online Campaigns

X_traditional = traditional_df[['MarketSize', 'AgeOfStore', 'Promotion', 'Week']]

y_traditional = traditional_df['SalesInThousands']

X_online = online_df[['impressions', 'CPC', 'CPM', 'reach', 'conversions']]

y_online = online_df['ctr']

# Split Data into Training and Testing Sets

X_train_trad, X_test_trad, y_train_trad, y_test_trad = train_test_split(X_traditional,
y_traditional, test_size=0.2, random_state=42)

X_train_online, X_test_online, y_train_online, y_test_online = train_test_split(X_online, y_online,
test_size=0.2, random_state=42)

# Initialize Models

models = {

    'Decision Tree': DecisionTreeRegressor(),

    'Random Forest': RandomForestRegressor(),

    'Linear Regression': LinearRegression(),

    'KNN': KNeighborsRegressor(n_neighbors=3)}

# Function to Train, Predict and Evaluate Models

def predict_and_visualize(X_train, y_train, X_test, y_test, dataset_name, model_name):

    model = models[model_name]
```

```python
model.fit(X_train, y_train)  # Train model

predictions = model.predict(X_test)  # Predict test data

# Add predictions to the original DataFrame

if dataset_name == "Traditional Campaign":

    traditional_df['Predicted_Sales'] = model.predict(X_traditional)

else:

    online_df['Predicted_CTR'] = model.predict(X_online)

# Visualization: Sales by Promotion and Market Size (Traditional Campaign)

if dataset_name == "Traditional Campaign":

    plt.figure(figsize=(10, 6))

    sns.boxplot(x='MarketSize', y='Predicted_Sales', hue='Promotion', data=traditional_df)

    plt.title(f"{model_name} - Sales by Market Size and Promotion (Traditional Campaign)")

    plt.ylabel("Predicted Sales in Thousands")

    plt.xlabel("Market Size")

    plt.legend(title="Promotion")

    plt.show()

# Scatter plot: Impressions vs Clicks (Online Campaign)

if dataset_name == "Online Campaign":

    plt.figure(figsize=(10, 6))
```

```
        plt.scatter(online_df['impressions'], online_df['clicks'], color='blue', label='Actual Clicks')

        plt.scatter(online_df['impressions'], online_df['Predicted_CTR'], color='red', label='Predicted
CTR')

        plt.title(f"{model_name} - Impressions vs Clicks (Online Campaign)")

        plt.xlabel("Impressions")

        plt.ylabel("Clicks / CTR")

        plt.legend()

        plt.show()

    # Box Plot: Sales Distribution by Market Size

    if dataset_name == "Traditional Campaign":

        plt.figure(figsize=(10, 6))

        sns.boxplot(x='MarketSize', y='Predicted_Sales', data=traditional_df)

        plt.title(f"{model_name} - Sales Distribution by Market Size (Traditional Campaign)")

        plt.ylabel("Predicted Sales in Thousands")

        plt.xlabel("Market Size")

        plt.show()

    # Box Plot: Sales Distribution by Promotion Type

    if dataset_name == "Traditional Campaign":

        plt.figure(figsize=(10, 6))

        sns.boxplot(x='Promotion', y='Predicted_Sales', data=traditional_df)
```

```
        plt.title(f"{model_name} - Sales Distribution by Promotion (Traditional Campaign)")

        plt.ylabel("Predicted Sales in Thousands")

        plt.xlabel("Promotion")

        plt.show()

# Function to evaluate model accuracy using RMSE and R² scores

def evaluate_model(X_train, y_train, X_test, y_test, model_name):

    model = models[model_name]

    model.fit(X_train, y_train)

    predictions = model.predict(X_test)

    mse = mean_squared_error(y_test, predictions)

    r2 = r2_score(y_test, predictions)

    rmse = np.sqrt(mse)

    return {'RMSE': rmse, 'R²': r2}

# Store results for all models

evaluation_results = {'Traditional Campaign': {}, 'Online Campaign': {}}

# Loop through each model and visualize predictions

for model_name in models.keys():

    print(f"\nVisualizing predictions for {model_name} (Traditional Campaign):")

    predict_and_visualize(X_train_trad, y_train_trad, X_test_trad, y_test_trad, "Traditional
Campaign", model_name)
```

```
    print(f"\nVisualizing predictions for {model_name} (Online Campaign):")

    predict_and_visualize(X_train_online, y_train_online, X_test_online, y_test_online, "Online Campaign", model_name)

    # Evaluate model performance

    trad_eval = evaluate_model(X_train_trad, y_train_trad, X_test_trad, y_test_trad, model_name)

    online_eval = evaluate_model(X_train_online, y_train_online, X_test_online, y_test_online, model_name)

    evaluation_results['Traditional Campaign'][model_name] = trad_eval

    evaluation_results['Online Campaign'][model_name] = online_eval

# Print evaluation results

print("\nModel Evaluation Results (Traditional Campaign):")

for model_name, metrics in evaluation_results['Traditional Campaign'].items():

    print(f"{model_name} - RMSE: {metrics['RMSE']}, R²: {metrics['R²']}")

print("\nModel Evaluation Results (Online Campaign):")

for model_name, metrics in evaluation_results['Online Campaign'].items():

    print(f"{model_name} - RMSE: {metrics['RMSE']}, R²: {metrics['R²']}")

# Conclusion: Identify the best model

def get_best_model(evaluation_results):

    best_model_trad = min(evaluation_results['Traditional Campaign'], key=lambda k: evaluation_results['Traditional Campaign'][k]['RMSE'])

    best_model_online = min(evaluation_results['Online Campaign'], key=lambda k:
```

```
evaluation_results['Online Campaign'][k]['RMSE'])

    print(f"\nConclusion: Best Model for Traditional Campaign is {best_model_trad} based on
RMSE.")

    print(f"Conclusion: Best Model for Online Campaign is {best_model_online} based on RMSE."

# Determine the best model

get_best_model(evaluation_results)
```

3.2 Website,visualisation and output

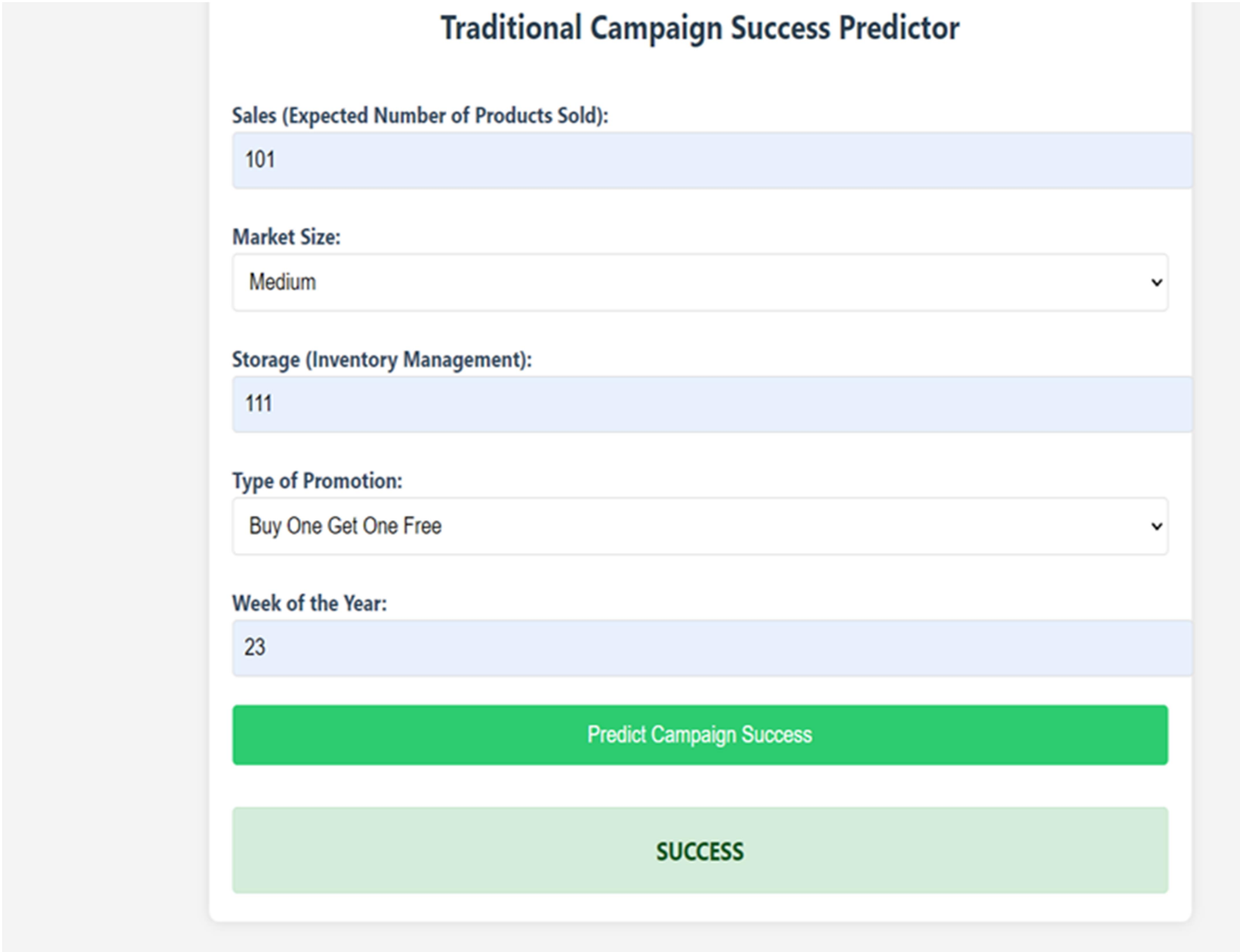
Evaluation Results

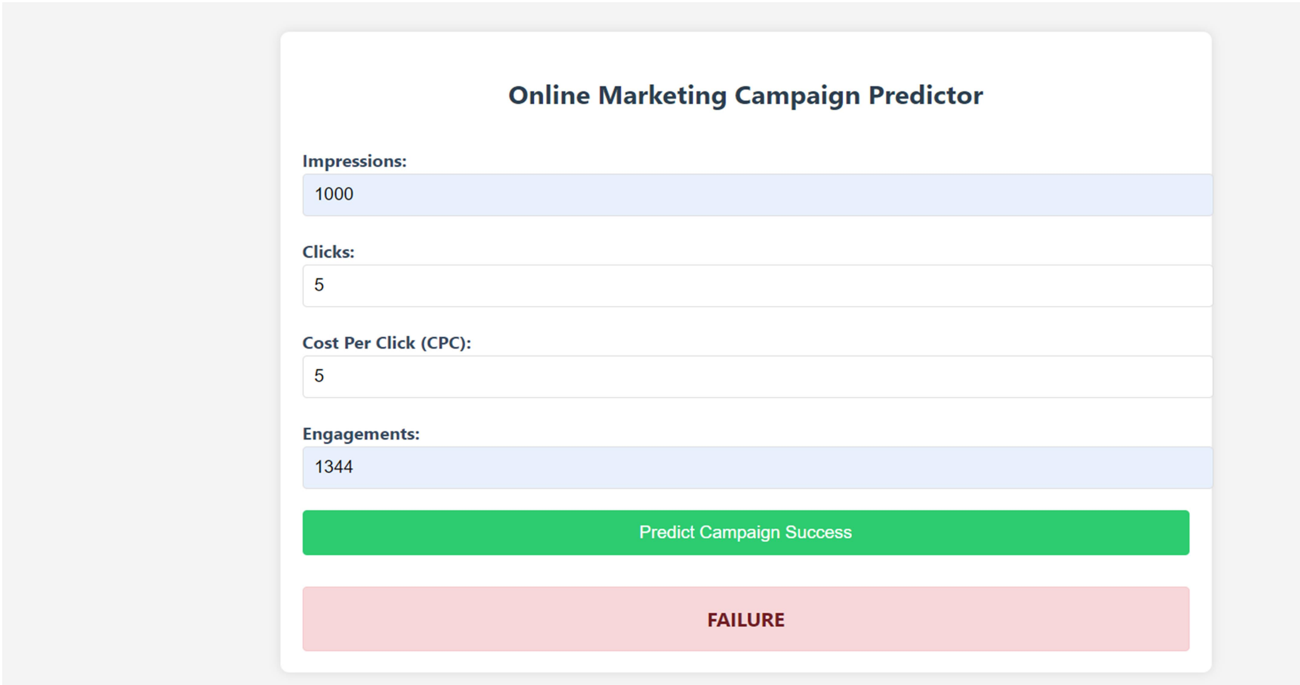
Traditional Campaign

Model	RMSE
Decision Tree	0.3999999999999986
Random Forest	0.9570000000000132
Linear Regression	0.6253588516746369
KNN	0.3333333333333215

Online Campaign

Model	RMSE
Decision Tree	0.0008333333333333318
Random Forest	0.00020277777777774966
Linear Regression	0.0007986106651210328
KNN	9.259259259259203e-05


Traditional Campaign Success Predictor
Sales (Expected Number of Products Sold):
101
Market Size:
Medium
Storage (Inventory Management):
111
Type of Promotion:
Buy One Get One Free
Week of the Year:
23
Predict Campaign Success
SUCCESS



CHAPTER 4

CONCLUSION

In conclusion, after evaluating four machine learning models, **K-Nearest Neighbors (KNN)** emerged as the most accurate model for both traditional and online marketing campaigns. This model had the lowest **RMSE** for predicting campaign performance, suggesting that it captures the underlying patterns in the data more effectively than the other models.

This system provides valuable insights for businesses, allowing them to optimize their marketing strategies based on data-driven predictions. By leveraging machine learning models like KNN, companies can improve their campaign effectiveness, allocate resources efficiently, and achieve better outcomes.

CHAPTER 5

BIBLIOGRAPHY

- Scikit-learn Documentation: https://scikit-learn.org/
- Matplotlib Visualization Guide: https://matplotlib.org/stable/contents.html
- Online and Traditional Marketing Overview: https://www.digitalmarketer.com/
- Pandas Data Manipulation Guide: https://pandas.pydata.org/