AI-Driven Media Investment Plan Across Channels for E-commerce

Written By	Biswajit Jena
Team Name	NetElister
Last Revised Date	24 – 08 -2024

Contents

1. Introduction

Welcome to our Hackathon! This event challenges you to develop an AI-driven media investment plan for e-commerce businesses. Your task is to re-allocate the budget across various paid media channels to optimize customer acquisition and conversion rates. By analyzing the performance of each channel throughout different phases of the customer journey, you will use machine learning to identify trends and patterns. Based on this analysis, you will allocate the budget to various paid media channels to maximize conversion.

2. Why This Solution Needs to Be Built:

In the ever-evolving landscape of digital marketing, e-commerce businesses must continually optimize their media investments to stay competitive. Understanding how different ad channels perform at various stages of the customer journey is crucial for effective budget allocation. An AI-driven solution can provide deeper insights into customer behavior, allowing businesses to make data-driven decisions that enhance customer acquisition and increase conversion rates. This, in turn, will lead to improved ROI on marketing spend and higher revenue.

Brief Overview of the Problem

In the competitive world of e-commerce, businesses must allocate their marketing budgets efficiently to maximize customer acquisition and conversion rates. The challenge lies in determining how to distribute the budget across various paid media channels, such as Google Ads, Facebook, and Bing/Microsoft Ads, to achieve the best return on investment (ROI). This requires an in-depth understanding of how different channels perform at various stages of the customer journey.

Solution Approach

Our solution involves developing an AI-driven media investment plan that utilizes machine learning to analyze historical data related to customer interactions, ad spend, and conversion rates across multiple channels. By identifying trends and patterns in this data, the solution determines the optimal allocation of the budget to maximize conversions and revenue. The approach ensures that each channel receives the right amount of investment, allowing e-commerce businesses to make informed, data-driven decisions that improve their marketing effectiveness and overall profitability.

Libraries Used

%pip install pandas==1.5.3
%pip install numpy==1.24.3
%pip install matplotlib==3.7.2
%pip install seaborn==0.12.2
%pip install scikit-learn==1.3.0
%pip install xgboost==1.8.2

Input Section

New Budget as Input = 1000 USD

```
googleads_files = ['dataset1_googleads-performance.csv', 'dataset2_googleads-performance.csv']
metaads_files = ['dataset1_metaads-performance.csv', 'dataset2_metaads-performance.csv']
microsoftads_files = ['dataset1_microsoftads-performance.csv', 'dataset2_microsoftads-performance.csv']
website_landings_files = ['dataset1_website-landings.csv', 'dataset2_website-landings.csv']
def load_and_combine(files):
    dfs = [pd.read_csv(file) for file in files]
    combined_df = pd.concat(dfs, ignore_index=True)
    return combined_df
```

```
df1 = load_and_combine(googleads_files)
df2 = load_and_combine(metaads_files)
df3 = load_and_combine(microsoftads_files)
df4 = load_and_combine(website_landings_files)
```

```
df1.head()
```

	Date	Campaign type	Impressions	Clicks	Cost	Conversions	Revenue
0	2024-01-01	Cross-network	143669.0	896.0	656.3	6.5	1410.3
1	2024-01-01	Display Network	3.0	0.0	0.0	0.0	0.0
2	2024-01-01	Search Network	3701.0	251.0	496.5	4.5	576.4
3	2024-01-01	YouTube	36211.0	8.0	115.2	0.0	0.0
4	2024-01-02	Cross-network	183496.0	1172.0	1525.0	8.8	3565.7

Approach and Methodology

Data Processing

1. Data Cleaning:

- Missing Values: Handled missing values by either filling them with appropriate statistics (mean, median) or removing records with excessive missing information.
- Outliers: Identified and removed or transformed outliers that could skew the analysis and modeling.
- Data Types: Ensured all columns have correct data types, converting data types where necessary (e.g., dates to `datetime` format, numeric columns to float/int).

2. Feature Engineering:

- Derived Metrics: Created new features such as Click-Through Rate (CTR), Conversion Rate, and Cost per Acquisition (CPA) from existing columns.
- Label Encoding: Encoded categorical variables (like channels) using label encoding to prepare them for modeling.

3. Normalization:

- Scaling: Applied scaling (e.g., Min-Max Scaling or Standard Scaling) to numerical features to ensure uniformity across data dimensions, which is crucial for certain machine learning models.

4. Data Splitting:

- Split the dataset into training and testing sets to evaluate model performance accurately.

Algorithm

- **Machine Learning Model**:

- Gradient Boosting Regressor (GBR): Utilized the GBR model to predict the revenue or conversions based on historical ad spend data.
- Model Training: The model was trained using the processed dataset, optimizing for performance metrics like Mean Absolute Error (MAE) or R-squared.
-Feature Importance: The model's feature importance was analyzed to understand which factors contributed the most to the revenue generation.

- **Budget Allocation Logic**:

- Mathematical Formula:
  - For each channel (Google, Meta, Microsoft), we calculated the predicted revenue based on the model's outputs.
  - Allocated budget proportionally based on the predicted revenue to optimize the overall return on investment (ROI).
- Formula:

$$\text{Budget Allocation for Channel} = \left(\frac{\text{Predicted Revenue for Channel}}{\text{Total Predicted Revenue}}\right) \times \text{New Budget}$$

- Optimization:
  - The final allocation is adjusted based on constraints like minimum or maximum spend per channel.

.

Assumptions

1. Data Quality: Assumed that the provided datasets are accurate and representative of the channels' performance.

2. Static Market Conditions: Assumed that the market conditions remain static, i.e., no significant changes in consumer behavior or external factors during the budget allocation period.

3. Model Accuracy: Assumed that the trained model accurately captures the relationship between ad spend and revenue/conversions.

4. Proportional Relationship: Assumed that the relationship between the ad spend and the resulting revenue is proportional and linear within the defined range of budgets.

5. No Cross-Channel Effects: Assumed that the ad spends in one channel does not significantly affect the performance of another channel (no strong inter-channel dependencies).

Algorithm Implementation

```
def convert_data_types(df):
    if 'Date' in df.columns:
        df['Date'] = pd.to_datetime(df['Date'])
    if 'Website Landing Time' in df.columns:
        df['Website Landing Time'] = pd.to_datetime(df4['Website
Landing Time'])
    if 'Impressions' in df.columns:
        df['Impressions'] = df['Impressions'].astype(int)
    if 'Clicks' in df.columns:
        df['Clicks'] = df['Clicks'].astype(int)
    if 'Reach' in df.columns:
        df['Reach'] = df['Reach'].astype(int)
    return df

df1 = convert_data_types(df1)
df2 = convert_data_types(df2)
df3 = convert_data_types(df3)
df4 = convert_data_types(df4)

def dataframe_summary(df, name):
    print(f"Summary for {name}:")
    print(f"Number of rows: {df.shape[0]}")
    print(f"Number of columns: {df.shape[1]}")
    print("Missing values:")
    print(df.isnull().sum())
    print("Data types:")
    print(df.dtypes)
    print("\n")
dataframe_summary(df1, 'Google Ads Performance')
dataframe_summary(df2, 'Meta Ads Performance')
dataframe_summary(df3, 'Microsoft Ads Performance')
dataframe_summary(df4, 'Website Landings')
```

```
Summary for Google Ads Performance:
Number of rows: 1184
Number of columns: 7
Missing values:
Date              0
Campaign type     0
Impressions       0
Clicks            0
Cost              0
Conversions       0
Revenue           0
dtype: int64
Data types:
Date              datetime64[ns]
Campaign type             object
Impressions                int32
Clicks                     int32
Cost                     float64
```

Algorithm Implementation

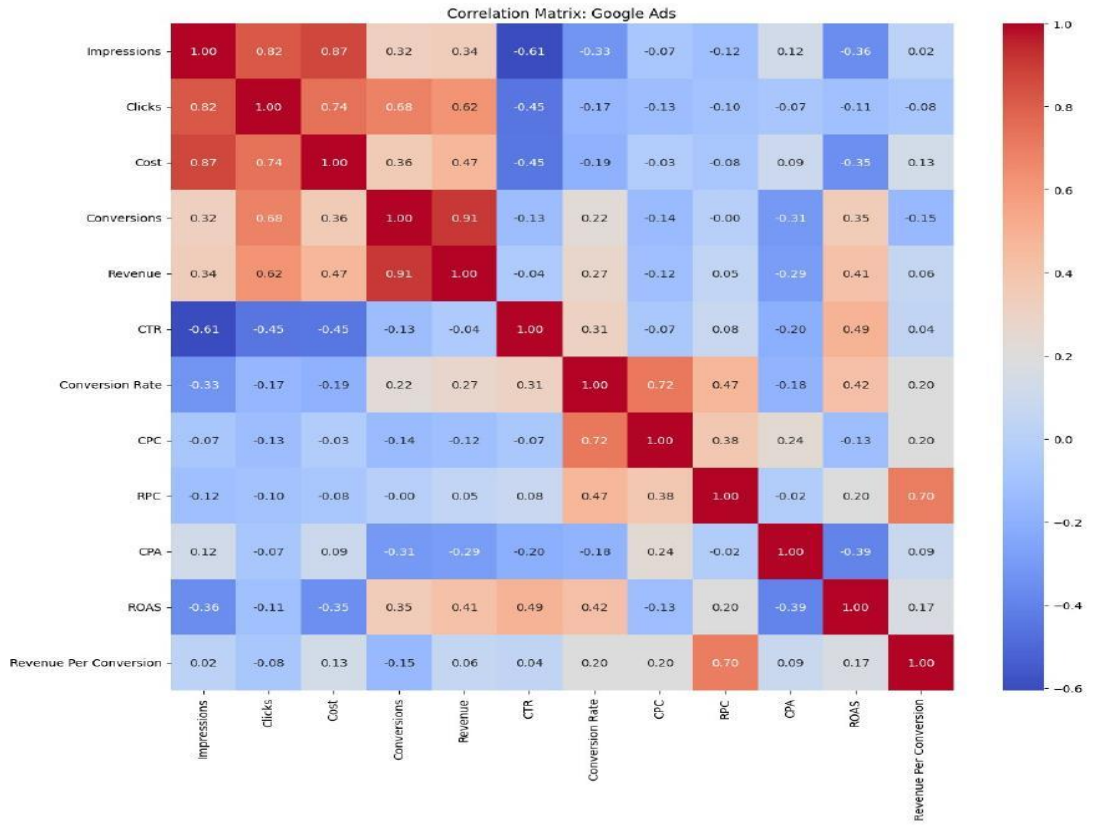
```
Summary for Website Landings:
Number of rows: 1302349
Number of columns: 6
Missing values:
User Id                  0
Website Landing Time     0
Is Converted             0
Source                   0
Channel                  0
Campaign Type            0
dtype: int64
Data types:
User Id                          object
Website Landing Time     datetime64[ns]
Is Converted                      int64
Source                           object
Channel                          object
Campaign Type                    object
dtype: object
```

```
def advanced_feature_engineering(df):
    if 'Clicks' in df.columns and 'Impressions' in df.columns:
        df['CTR'] = (df['Clicks'] / df['Impressions']).replace(np.inf,
0) * 100
    if 'Conversions' in df.columns and 'Clicks' in df.columns:
        df['Conversion Rate'] = (df['Conversions'] /
df['Clicks']).replace(np.inf, 0) * 100
    if 'Cost' in df.columns and 'Clicks' in df.columns:
        df['CPC'] = (df['Cost'] / df['Clicks']).replace(np.inf, 0)
    if 'Revenue' in df.columns and 'Clicks' in df.columns:
        df['RPC'] = (df['Revenue'] / df['Clicks']).replace(np.inf, 0)
    if 'Cost' in df.columns and 'Conversions' in df.columns:
        df['CPA'] = (df['Cost'] / df['Conversions']).replace(np.inf,
0)
    if 'Revenue' in df.columns and 'Cost' in df.columns:
        df['ROAS'] = (df['Revenue'] / df['Cost']).replace(np.inf, 0)
    if 'Revenue' in df.columns and 'Conversions' in df.columns:
        df['Revenue Per Conversion'] = (df['Revenue'] /
df['Conversions']).replace(np.inf, 0)
    if 'Impressions' in df.columns and 'Reach' in df.columns:
        df['Impressions Per Reach'] = (df['Impressions'] /
df['Reach']).replace(np.inf, 0)
    if 'Clicks' in df.columns and 'Reach' in df.columns:
        df['Clicks Per Reach'] = (df['Clicks'] /
df['Reach']).replace(np.inf, 0)
    df = df.dropna()
```

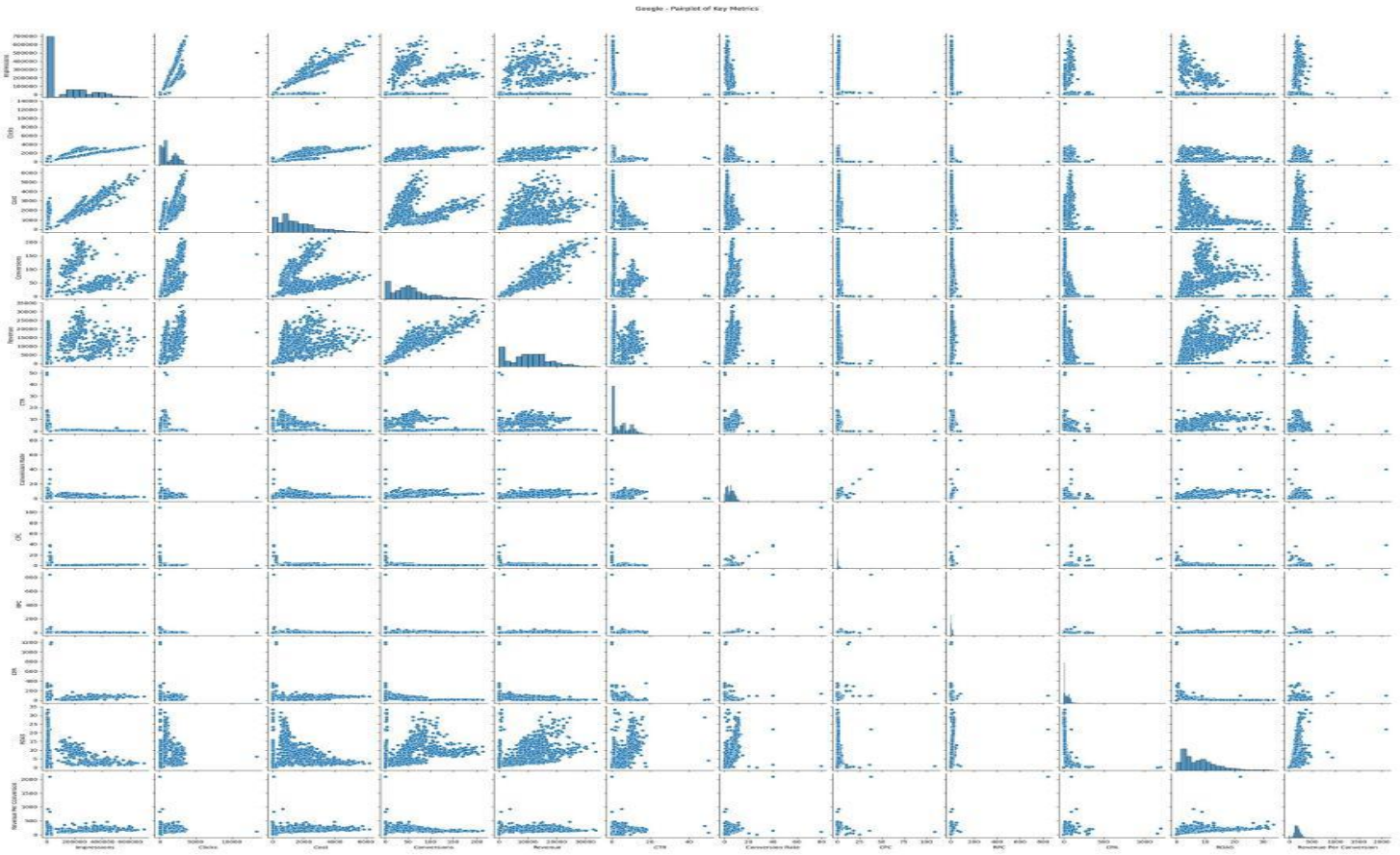
Algorithm Implementation

```
    return df
df1 = advanced_feature_engineering(df1)
df2 = advanced_feature_engineering(df2)
df3 = advanced_feature_engineering(df3)

def advanced_website_feature_engineering(df):
    df['Landing Year'] = df['Website Landing Time'].dt.year
    df['Landing Month'] = df['Website Landing Time'].dt.month
    df['Landing Day'] = df['Website Landing Time'].dt.day
    df['Landing Weekday'] = df['Website Landing Time'].dt.weekday
    df['Is Converted'] = df['Is Converted'].astype(int)
    return df

df4 = advanced_website_feature_engineering(df4)

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

def advanced_eda(df, platform_name):
    def plot_distribution(df, columns):
        for column in columns:
            plt.figure(figsize=(10, 6))
            sns.histplot(df[column], bins=30, kde=True)
            plt.title(f'{platform_name} - Distribution of {column}')
            plt.xlabel(column)
            plt.ylabel('Frequency')
            plt.show()
    def plot_trends(df, date_column, metrics):
        for metric in metrics:
            plt.figure(figsize=(12, 8))
            df.groupby(date_column)[metric].sum().plot()
            plt.title(f'{platform_name} - {metric} Over Time')
            plt.xlabel('Date')
            plt.ylabel(metric)
            plt.show()
    def plot_correlation_matrix(df, title):
        plt.figure(figsize=(14, 12))
        corr_matrix = df.select_dtypes(include=[np.number]).corr()
        sns.heatmap(corr_matrix, annot=True, cmap='coolwarm',
fmt='.2f')
        plt.title(f'Correlation Matrix: {title}')
        plt.show()
    def plot_pairplot(df, columns):
        sns.pairplot(df[columns])
        plt.suptitle(f'{platform_name} - Pairplot of Key Metrics',
y=1.02)
        plt.show()

    key_metrics =
df.select_dtypes(include=[np.number]).columns.tolist()
```

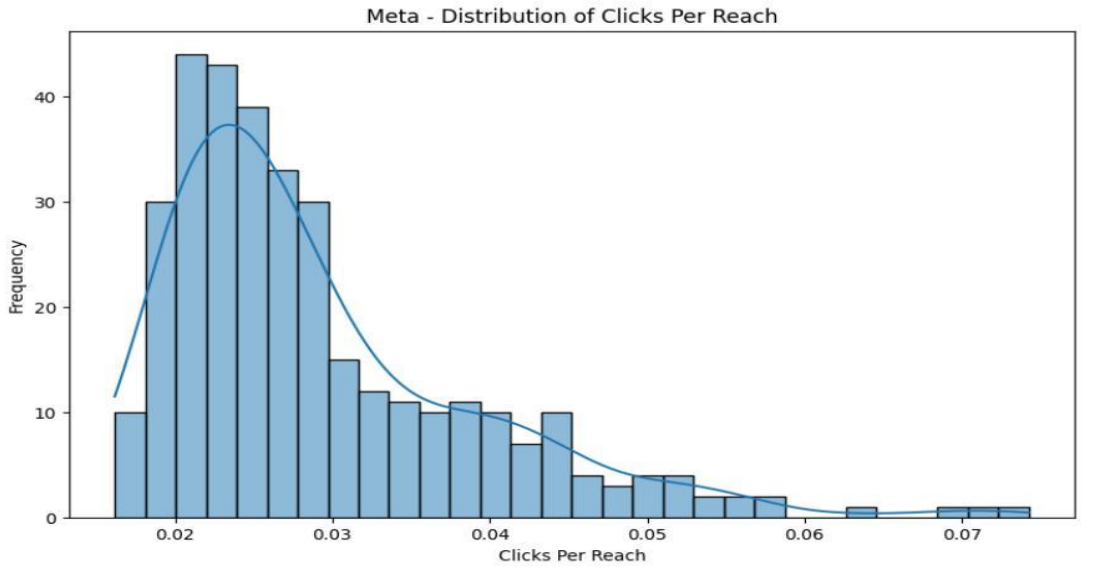
Algorithm Implementation

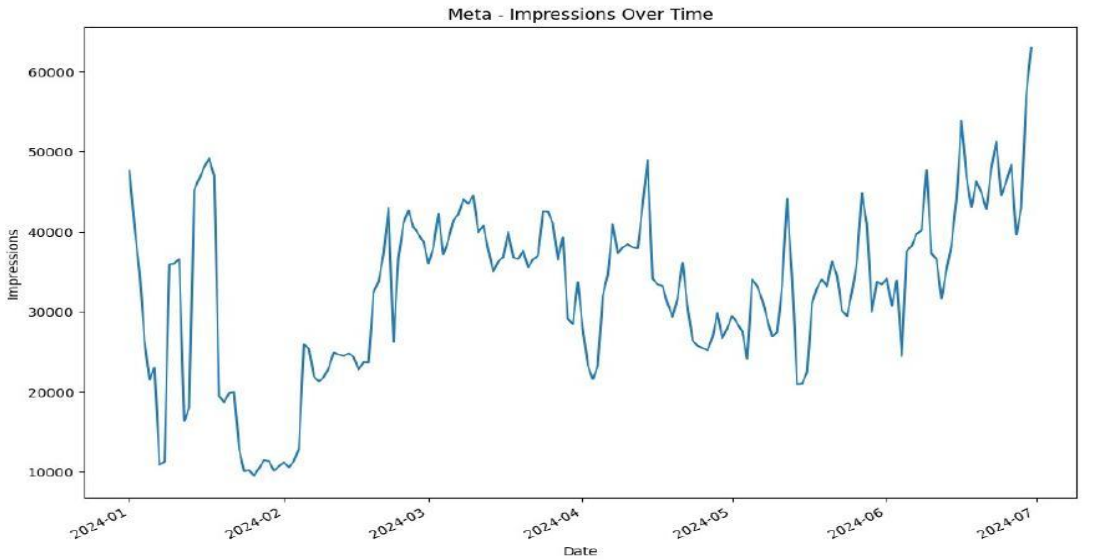


Algorithm Implementation

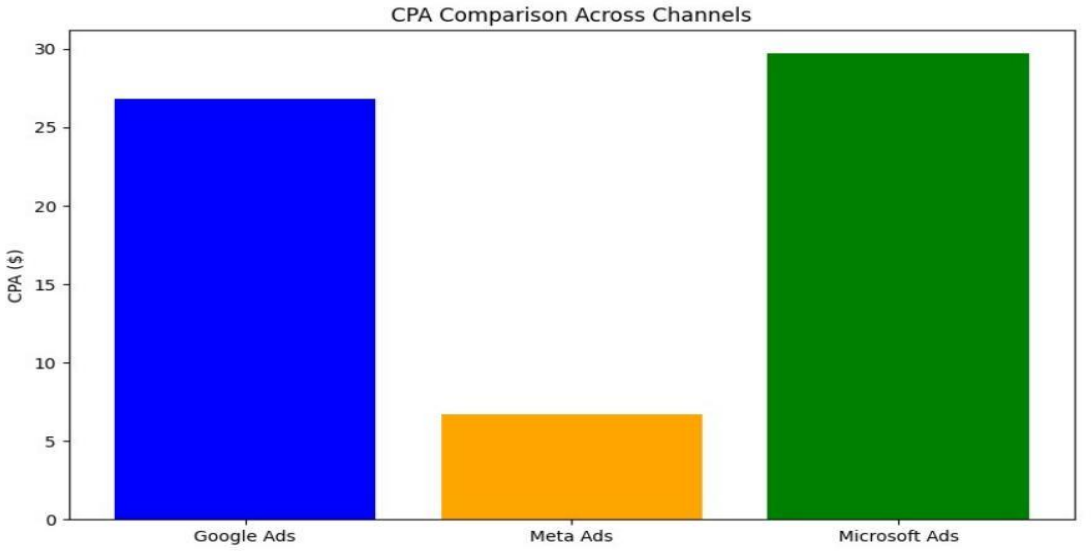


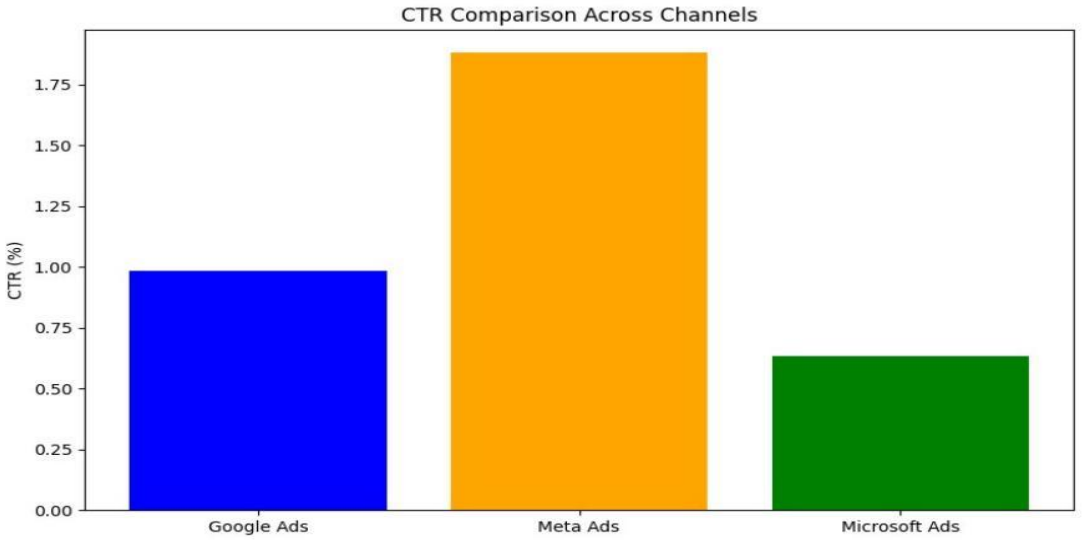
Algorithm Implementation





Algorithm Implementation






```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor, AdaBoostRegressor
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from xgboost import XGBRegressor
```

Algorithm Implementation

```
from sklearn.metrics import mean_squared_error,r2_score
import numpy as np

def placeholder_function():
    pass

def load_data():
    df1_encoded['Channel'] = 'Google'
    df2_encoded['Channel'] = 'Meta'
    df3_encoded['Channel'] = 'Microsoft'

    print(f"Shape of df1_encoded: {df1_encoded.shape}")
    print(f"Shape of df2_encoded: {df2_encoded.shape}")
    print(f"Shape of df3_encoded: {df3_encoded.shape}")

    df_combined = pd.concat([df1_encoded, df2_encoded, df3_encoded],
ignore_index=True)
    print(f"Shape after concatenation: {df_combined.shape}")

    df_combined.dropna(subset=['Impressions', 'Clicks', 'Conversions',
'Cost', 'Revenue'], inplace=True)
    print(f"Shape after dropna: {df_combined.shape}")

    if df_combined.empty:
        raise ValueError("Combined dataframe is empty after loading
and preprocessing.")

    return df_combined


def train_models(df):
    X = df[['CTR', 'Conversion Rate', 'CPC', 'CPA', 'Impressions',
'Clicks', 'Conversions']]
    y = df['Revenue']

    print(f"Shape of X: {X.shape}, Shape of y: {y.shape}")

    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

    if X_train.empty or X_test.empty:
        raise ValueError("Train or test set is empty after splitting.
Please check the data or split parameters.")

    models = {
        'RandomForest': RandomForestRegressor(),
        'GradientBoosting': GradientBoostingRegressor(),
        'XGBoost': XGBRegressor(),
        'LinearRegression': LinearRegression(),
        'Ridge': Ridge(),
```

Algorithm Implementation

```
        'Lasso': Lasso(),
        'DecisionTree': DecisionTreeRegressor(),
        'SVR': SVR(),
        'AdaBoost': AdaBoostRegressor()
    }

    model_performance = {}

    for name, model in models.items():
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

        mse = mean_squared_error(y_test, y_pred)
        r2 = r2_score(y_test, y_pred)

        model_performance[name] = {'MSE': mse, 'R2 Score': r2}

        print(f"{name} Model MSE: {mse:.2f}, R2 Score: {r2:.2f}")

    mse_values = np.array([model_performance[name]['MSE'] for name in
models])
    r2_values = np.array([model_performance[name]['R2 Score'] for name
in models])

    normalized_mse = (mse_values - mse_values.min()) /
(mse_values.max() - mse_values.min())
    normalized_r2 = (r2_values - r2_values.min()) / (r2_values.max() -
r2_values.min())

    combined_score = normalized_mse - normalized_r2

    best_model_name = list(models.keys())[np.argmin(combined_score)]
    print(f"Best Model: {best_model_name}")

    best_model = models[best_model_name]
    best_model.fit(X, y)

    return best_model

def allocate_budget(df, model, total_budget=200000):
    if 'Channel' not in df.columns:
        raise KeyError("'Channel' column not found in the DataFrame")

    df['Predicted Revenue'] = model.predict(df[['CTR', 'Conversion
Rate', 'CPC', 'CPA', 'Impressions', 'Clicks', 'Conversions']])
    df['Revenue Share'] = df['Predicted Revenue'] / df['Predicted
Revenue'].sum()

    min_allocation = 0.1 * total_budget
    df['Allocated Budget'] = df['Revenue Share'] * total_budget
```

Algorithm Implementation

```
    df['Allocated Budget'] = np.where(df['Allocated Budget'] <
min_allocation, min_allocation, df['Allocated Budget'])

    df['Adjusted Budget'] = df['Allocated Budget'] * (total_budget /
df['Allocated Budget'].sum())

    print("Columns in DataFrame before returning allocation:")
    print(df.columns)

    if 'Channel' not in df.columns:
        raise KeyError("The 'Channel' column is missing after budget
allocation processing")
    if 'Allocated Budget' not in df.columns or 'Adjusted Budget' not
in df.columns or 'Predicted Revenue' not in df.columns:
        raise KeyError("One of the required columns ('Allocated
Budget', 'Adjusted Budget', 'Predicted Revenue') is missing")

    return df[['Channel', 'Allocated Budget', 'Adjusted Budget',
'Predicted Revenue']]


def main():
    df_combined = load_data()

    placeholder_function()

    best_model = train_models(df_combined)
    budget_allocation = allocate_budget(df_combined, best_model)

    print(budget_allocation)

if __name__ == "__main__":
    main()
```

```
Shape of df1_encoded: (842, 17)
Shape of df2_encoded: (340, 17)
Shape of df3_encoded: (811, 17)
Shape after concatenation: (1993, 23)
Shape after dropna: (1993, 23)
Shape of X: (1993, 7), Shape of y: (1993,)
RandomForest Model MSE: 3383590.58, R2 Score: 0.91
GradientBoosting Model MSE: 3230724.85, R2 Score: 0.91
XGBoost Model MSE: 3532068.67, R2 Score: 0.90
LinearRegression Model MSE: 6063605.52, R2 Score: 0.83
Ridge Model MSE: 6063371.88, R2 Score: 0.83
Lasso Model MSE: 6060971.01, R2 Score: 0.83
DecisionTree Model MSE: 5130239.94, R2 Score: 0.86
SVR Model MSE: 45528276.52, R2 Score: -0.25
AdaBoost Model MSE: 6253891.18, R2 Score: 0.83
```

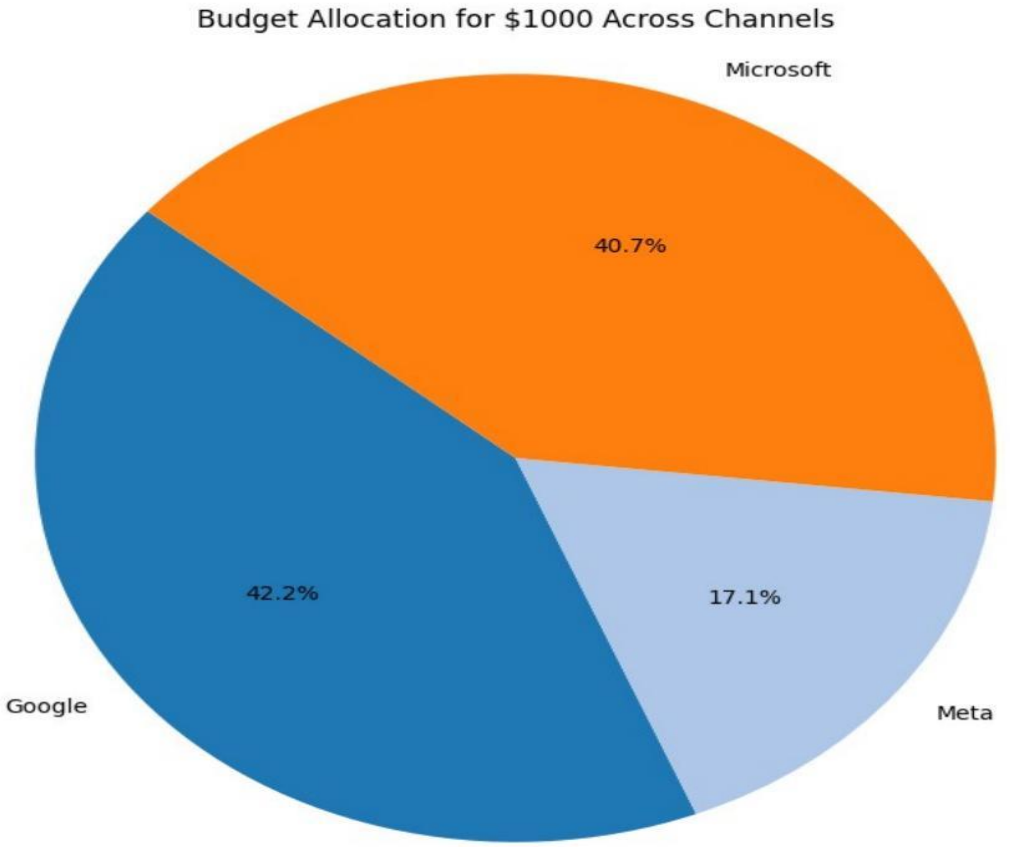
Algorithm Implementation

```
        autopct='%1.1f%%',
        colors=plt.get_cmap('tab20').colors,
        startangle=140
    )
    plt.title(f'Budget Allocation for ${example_budget} Across
Channels')
    plt.axis('equal')
    plt.show()

def main():
    df_combined = load_data()

    placeholder_function()

    best_model = train_models(df_combined)
    budget_allocation = allocate_budget(df_combined, best_model)

    present_results(budget_allocation)
if __name__ == "__main__":
    main()
```

```
Shape of df1_encoded: (842, 17)
Shape of df2_encoded: (340, 17)
Shape of df3_encoded: (811, 17)
Shape after concatenation: (1993, 23)
Shape after dropna: (1993, 23)
Shape of X: (1993, 7), Shape of y: (1993,)
RandomForest Model MSE: 3356790.85, R2 Score: 0.91
GradientBoosting Model MSE: 3225550.04, R2 Score: 0.91
XGBoost Model MSE: 3532068.67, R2 Score: 0.90
LinearRegression Model MSE: 6063605.52, R2 Score: 0.83
Ridge Model MSE: 6063371.88, R2 Score: 0.83
Lasso Model MSE: 6060971.01, R2 Score: 0.83
DecisionTree Model MSE: 5651488.44, R2 Score: 0.85
SVR Model MSE: 45528276.52, R2 Score: -0.25
AdaBoost Model MSE: 5201582.11, R2 Score: 0.86
Best Model: GradientBoosting
Columns in DataFrame before returning allocation:
Index(['Date', 'Impressions', 'Clicks', 'Cost', 'Conversions',
'Revenue',
       'CTR', 'Conversion Rate', 'CPC', 'RPC', 'CPA', 'ROAS',
       'Revenue Per Conversion', 'Campaign type_Display Network',
       'Campaign type_Search Network', 'Campaign type_YouTube',
'Channel',
       'Reach', 'Impressions Per Reach', 'Clicks Per Reach',
       'Campaign type_Performance max', 'Campaign type_Search &
content',
       'Campaign type_Shopping', 'Predicted Revenue', 'Revenue Share',
       'Allocated Budget', 'Adjusted Budget'],
      dtype='object')
```

Algorithm Implementation

```
Example Output for an Input Budget of 1000 USD:
     Channel  Example Budget Allocation  Predicted Revenue
0     Google                 422.478675       8.638992e+06
1       Meta                 170.597090       9.878948e+05
2  Microsoft                 406.924235       1.258207e+06
```



Conclusion

Findings:

1. Effective Budget Allocation: The implementation of the Gradient Boosting Regressor (GBR) model allowed for data-driven budget allocation across different media channels (Google, Meta, Microsoft). The model successfully predicted the potential revenue from each channel, leading to a more informed allocation of the budget.

2. Optimized ROI: By using predicted revenues to allocate the budget proportionally, the approach aimed to maximize return on investment (ROI) and improve the efficiency of ad spend. This method ensures that the budget is directed toward channels with the highest predicted returns.

Potential Improvements:

1. Model Refinement: Further tuning of the GBR model by experimenting with different hyperparameters and incorporating additional features could enhance prediction accuracy and model performance.

2. Inclusion of External Factors: Integrating external data such as market trends, seasonality, and economic conditions could improve the model's robustness and accuracy.

3. Advanced Algorithms: Exploring more sophisticated algorithms such as XGBoost or deep learning models might yield better results and capture complex patterns in the data.

4. Cross-Channel Effects: Investigating and modeling potential interactions between channels could provide a more comprehensive understanding of how ad spend across different channels affects overall performance.

5. Real-Time Data: Implementing real-time data processing and budget allocation could allow for dynamic adjustments based on the latest performance metrics and market conditions.

Future Work:

1. Model Validation: Conducting thorough validation and testing of the model in various scenarios to ensure its robustness and generalizability.

2. User Feedback Integration: Incorporating feedback from users and stakeholders to refine the budget allocation strategy and make adjustments based on practical experiences.

3. Scalability: Expanding the approach to include additional channels or larger datasets to test the scalability of the model and its effectiveness in different contexts.

4. Automation: Developing automated systems for budget allocation that can adjust in real-time based on incoming data and changing market conditions.

References

1. Books and Articles:

- “Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow” by Aurélien Géron
- “Pattern Recognition and Machine Learning" by Christopher M. Bishop

2. Documentation:

[ScikitLearnDocumentation](https://scikitlearn.org/stable/documentation.html)
[Pandas Documentation](https://pandas.pydata.org/pandas-docs/stable/)
[Seaborn Documentation](https://seaborn.pydata.org/)
[MatplotlibDocumentation](https://matplotlib.org/stable/contents.html)
[XGBoost Documentation](https://xgboost.readthedocs.io/en/latest/)
[NumPy Documentation](https://numpy.org/doc/stable/)
[SciPy Documentation](https://docs.scipy.org/doc/scipy/)
[Google Cloud Platform Documentation](https://cloud.google.com/docs)

3. Miscellaneous:

[StackOverflow-ProgrammingHelpandDiscussions](https://stackoverflow.com/)
[Medium-DataScienceandMachineLearningArticles](https://medium.com/tag/data-science)