Data Collection - Gathering Data
The raw data comes from the many campaigns conducted like google ads and facebook ads previously conducted campaigns.

Identifying Data Sources:

- Google Ads (Impressions, Clicks, CPC, CTR, Conversions)
- Facebook Ads (Engagement, Clicks, Conversions)
- Website Analytics (Bounce Rate, Session Duration, Returning Users)
- CRM Data (Customer details, past purchases, lifetime value)

Extracting & Storing Data:

- Pulling data from APIs (Google Analytics, Facebook Ads Manager)
- Storing it in a structured database (PostgreSQL or BigQuery)

Checking Data Quality:

- Ensuring there are no missing values, duplicates, or inconsistencies.This will be done EDA part

Here the simple data looks like

Metric	**Impressions**	**Clicks**	**CTR (%)**	**Conversions**	**CPC ($)**	**CPA ($)**	**Session Duration (sec)**	**Bounce Rate (%)**
Mean	6,700	242	3.4	9.2	0.7	62.6	110	44
Min	3,000	90	2.0	3	0.5	25.0	80	30
Max	10,000	500	5.0	20	0.9	90.0	150	55

But in reality this can be more and more huge and there are many metrics we can get.

EDA(Exploratory Data Analysis)
Summarizing the Data:

- Checking **mean, median, min, max, standard deviation** for key metrics like CPC, CTR, and conversions.

Visualizing Trends:

- **Histogram** for CPC to understand cost distribution.
- **Scatter plot** of Bounce Rate vs. Session Duration to see correlations.
- **Bar chart** of Conversion Rate per Traffic Source.

Detecting Anomalies:

- Outliers in CPC (e.g., a campaign with $50 CPC is an anomaly).
- Extremely low engagement sessions (Session Duration < 5s).

Data Preprocessing - Cleaning the Data

1. **Handling Missing Data:**

   Filling missing CPC values with the **median CPC**.

   Assigning "Unknown" to missing Traffic Source values.

2. **Handling Outliers:**

   Removing CPC values > $5 (too high for most digital ads).

   Flagging users with **Session Duration < 5s** as "Low Engagement".

3. **Encoding Categorical Variables:**

   Converting **Traffic Source (Google, Facebook, etc.)** to **one-hot encoding**.

   Label encoding for **Returning User (Yes/No → 1/0)**.

4. **Scaling Numerical Features:**

   Standardizing CPC, Session Duration, Bounce Rate using **MinMaxScaler**.

Feature Engineering - Creating New Features
Creating New Features for Better Insights:

- **Ad Efficiency Score:** Ad_Efficiency=CTR×Conversion_RateCPCAd_Efficiency = \frac{CTR \times Conversion_Rate}{CPC}Ad_Efficiency=CPCCTR×Conversion_Rate
- **Engagement Score:** Engagement_Score=Session_DurationBounce_Rate×Returning_UserEngagement_Score = \frac{Session_Duration}{Bounce_Rate} \times Returning_UserEngagement_Score=Bounce_RateSession_Duration×Returning_User
- **Customer Value Prediction (CLV):** CLV=Average_Purchase_Value×Purchase_Frequency×Customer_LifespanCLV = Average_Purchase_Value \times Purchase_Frequency \times Customer_LifespanCLV=Average_Purchase_Value×Purchase_Frequency×Customer_Lifespan

Transforming Features:

- Converting **time-based features** (e.g., timestamps → day of week, hour of day).
- Creating **interaction terms** (e.g., CPC × CTR → Cost-Adjusted CTR).

Feature Selection - Choosing the Best Features

Checking Feature Importance (Random Forest Method):

- Training a **Random Forest model on all features**.
- Extracting **importance scores** for each feature.
- Removing **low-importance features**.

Removing Highly Correlated Features:

- If **CTR and Clicks** are highly correlated, keep only one.

Applying Variance Thresholding:

- Removing features with **almost no variation**

Model Training - Building the Random Forest Model

Random Forest is a **supervised machine learning algorithm** that belongs to the ensemble learning category. It is built using multiple **decision trees** and makes predictions by aggregating the output from each tree. This approach helps in **reducing overfitting** and improves generalization.

It can be used for both **classification** (e.g., predicting whether a user will convert or not) and **regression** (e.g., predicting expected ad revenue per user).

Digital marketing data is often complex, noisy, and contains many variables (features). The **Random Forest model** is a great fit for this use case due to:

1. **Handling High-Dimensional Data**
    - Digital marketing campaigns generate massive datasets, including impressions, clicks, CTR, CPC, bounce rate, and conversions.
    - Random Forest efficiently selects important features without requiring extensive manual tuning.
2. **Reducing Overfitting**
    - Individual decision trees can overfit, but Random Forest, by averaging multiple trees, provides **robust and stable** predictions.
3. **Handling Missing and Noisy Data**
    - Many marketing datasets have missing values (e.g., users not clicking ads, missing demographic info).
    - Random Forest can handle missing data better than other models by using median values and feature importance.
4. **Feature Importance Insights**
    - It helps marketers **understand which factors influence conversions** the most (e.g., is CPC or engagement score more important?).
    - This is crucial for optimizing ad spend and targeting the right audience.

Predicting User Behavior: Will a user convert after clicking an ad?

Ad Spend Optimization: Helps in deciding which campaigns perform best.

Feature Importance: Identifies which marketing KPIs contribute the most to conversions.

Handling Noisy Data: Reduces errors due to missing or inconsistent data.

splitting Data:

- **80% for training, 20% for testing**.

Training the Random Forest Model:

- Using cleaned & selected features.
- Hyperparameter tuning (n_estimators, max_depth).

Evaluating Model Performance:

- Checking **accuracy, precision, recall, F1-score**.
- Comparing results with **other models (Logistic Regression, XGBoost)**.

Model Testing & Evaluation

1. **Measuring Performance:**
   - Using **ROC-AUC curve** to evaluate classification performance.
   - Checking **feature importance** to see which features impact predictions the most.
2. **Comparing with Baseline Model:**
   - If **Random Forest outperforms Logistic Regression**, we keep it.
3. **Analyzing Errors:**
   - Checking misclassified users to understand why the model made mistakes.

Model Deployment & Monitoring

1. **Deploying the Model in a Cloud Environment:**
   - Using **AWS, GCP, or Azure** for real-time predictions.
2. **Integrating with Ad Platforms:**
   - Feeding predictions into **Google Ads Manager** to optimize bidding strategies.
3. **Monitoring Model Performance:**
   - Setting up **drift detection** (if new data distribution changes, retrain the model).

The first step in MLOps is **data versioning and pipeline automation**. Since marketing data changes frequently (e.g., new ad campaigns, seasonal variations), we use a system like **DVC (Data Version Control)** to track changes in datasets. This ensures that we can always retrieve past versions of the data and train models on consistent datasets. Additionally, we create automated **data ingestion pipelines** that pull data from ad platforms (Google Ads, Facebook Ads) and preprocess it for training.

Once the data is managed, we implement **experiment tracking and model versioning**. In digital marketing, we often experiment with different models and hyperparameters to find the best-performing one. We use tools like **MLflow or Weights & Biases** to log every experiment, track changes, and compare model performance. This allows us to identify which Random Forest model configuration delivers the highest accuracy for conversion prediction.

After selecting the best model, we focus on **deployment**. The trained Random Forest model is deployed as an **API** using **FastAPI or Flask**, allowing marketing teams to send real-time ad data and receive predictions on conversion rates. The model runs on **AWS, GCP, or Azure**, ensuring scalability. To make deployment efficient, we use **Docker** for containerization and **Kubernetes** for managing multiple instances if the load increases.

Once deployed, the most critical part of MLOps is **monitoring and model retraining**. Since ad trends change frequently, the model's accuracy can degrade over time due to **data drift**. To monitor this, we set up **automated alerts** that track performance metrics such as accuracy, precision, and recall. If the performance drops below a threshold, the system triggers

automated retraining using newly collected marketing data. This ensures that our model remains effective without manual intervention.

Finally, to streamline the entire process, we implement **CI/CD (Continuous Integration and Continuous Deployment) pipelines**. Whenever there's a data update or a model improvement, GitHub Actions or Jenkins automatically retrain the model, validate performance, and deploy the updated version. This automation minimizes downtime and ensures that the best model is always in production.

By integrating **MLOps** into this project, we ensure that our digital marketing model is not only accurate but also **scalable, reliable, and adaptable** to changing trends. This allows businesses to make data-driven marketing decisions with confidence.