**CTR PREDICTION AND OPTIMIZATION**

The CTR (Click-Through Rate) Prediction and Optimization application predicts the CTR based on input features like city, time of day, preferred channel, type of device, and type of CTA (Call-to-Action). The application leverages machine learning to estimate the expected CTR based on historical data and returns the result through a web interface powered by React and Flask.

1. Objective:

Analyze the provided dataset of business owners programmatically using TensorFlow and a React-based API. Your goal is to build a machine learning model that predicts and optimizes the Click-Through Rate (CTR) based on various data attributes.

1. Technologies Used:

#### Backend:

* Flask: For creating the API and serving the model predictions.
* TensorFlow/Keras: For building and training the machine learning model.
* scikit-learn: For data preprocessing (encoding, scaling).

#### Frontend:

* React: For building the user interface, form submission, and API integration.
* CSS: For styling the form and components.

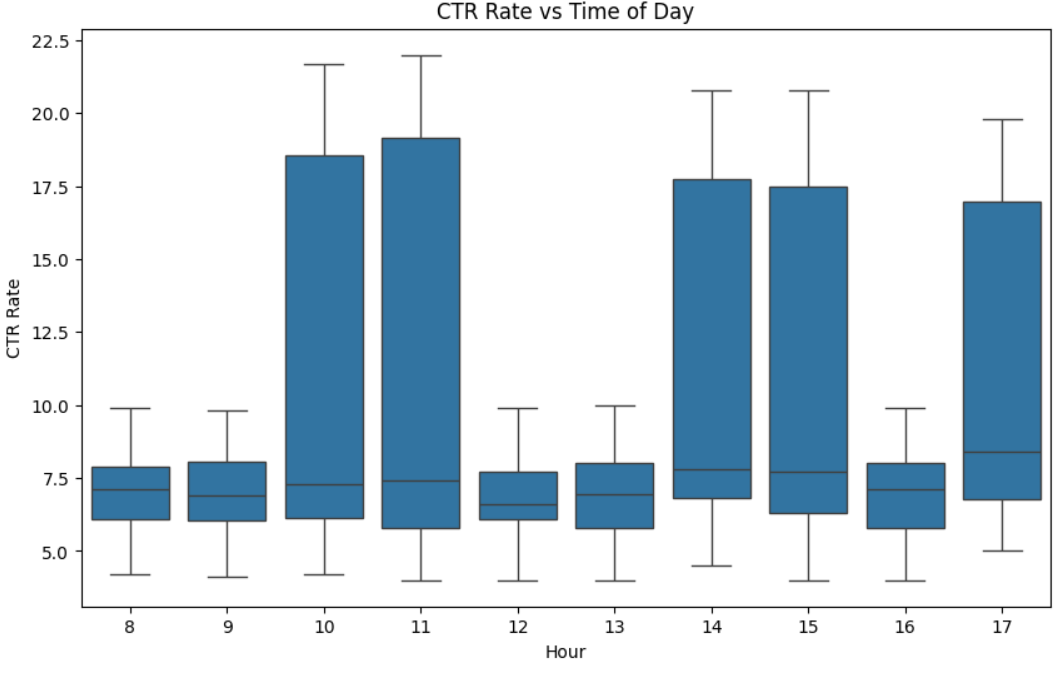
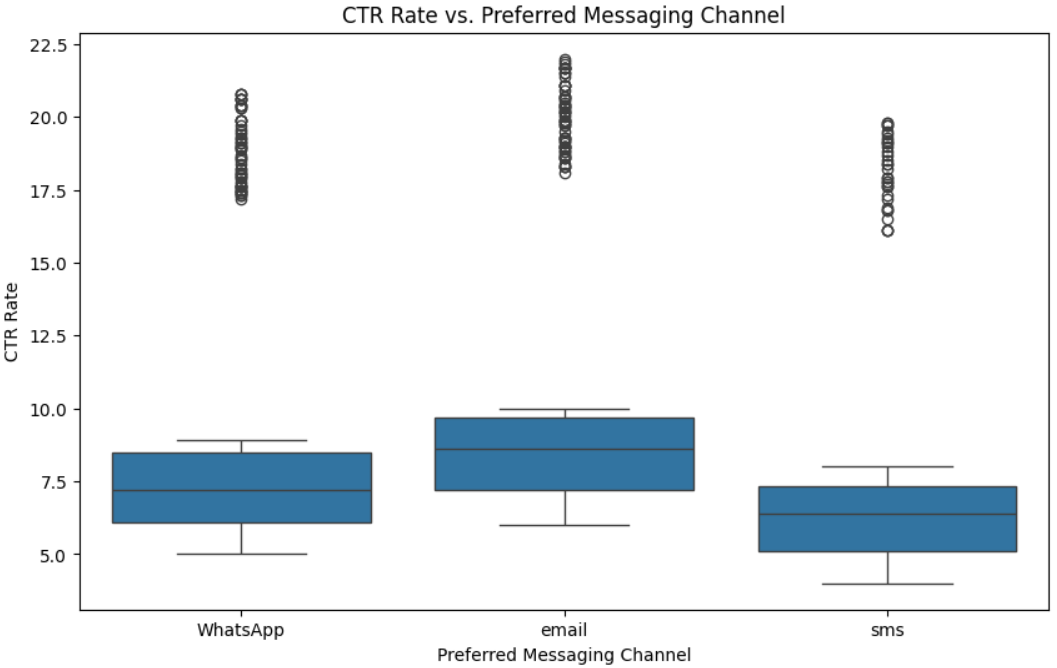
1. Process of App creation:

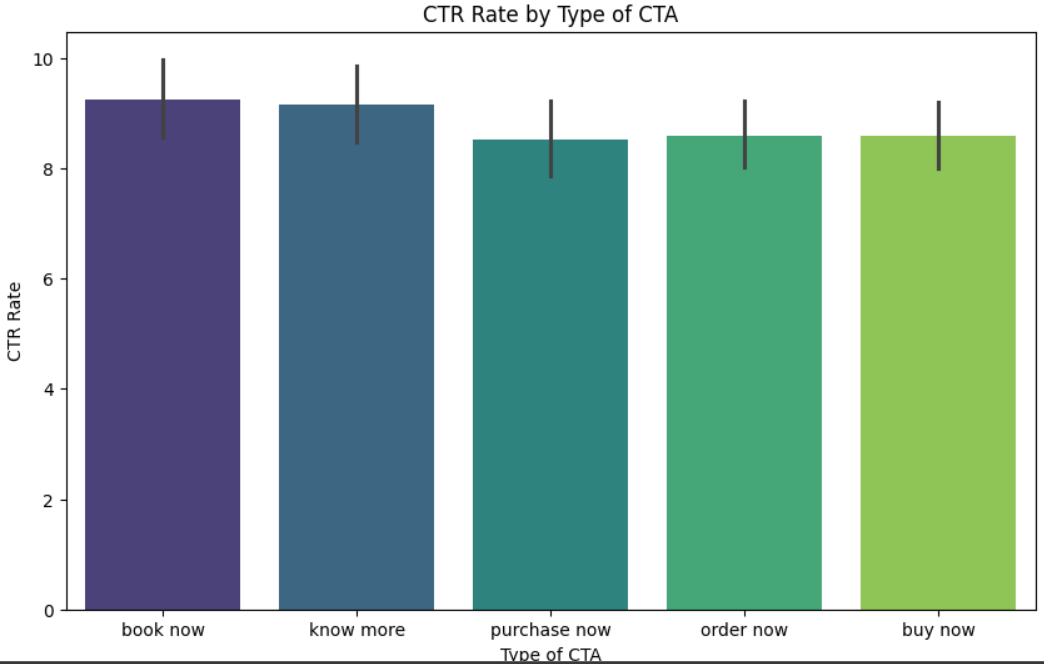
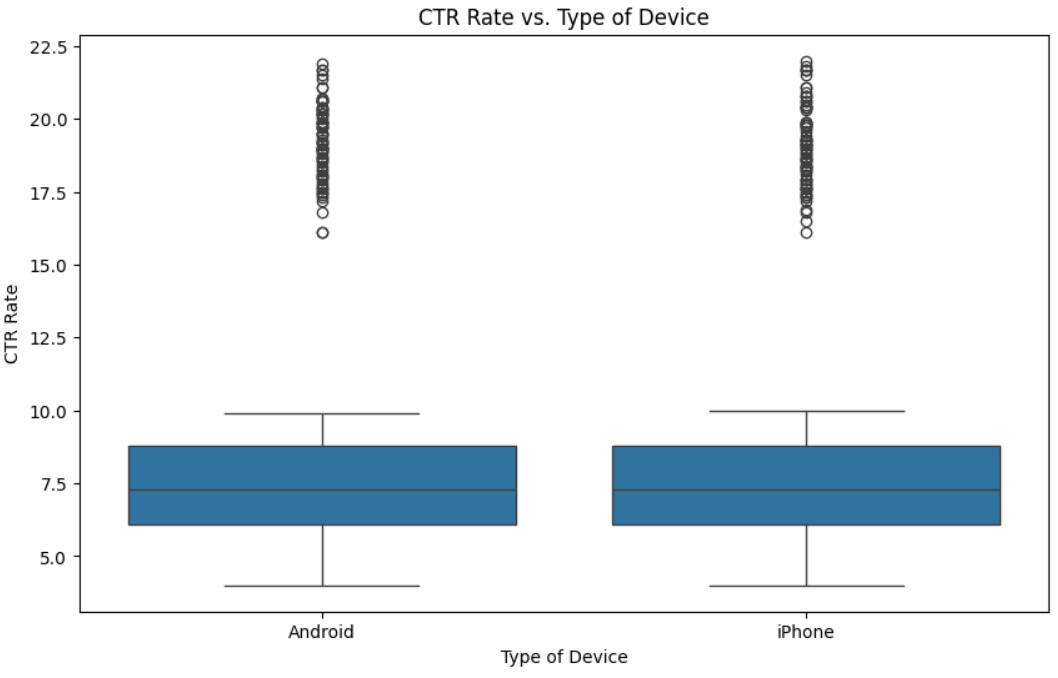
### **A. Model Creation and Prediction**

#### **Dataset Preparation**

* The dataset contains fields such as City, Preferred Messaging Channel, Type of Device, Type of CTA, Time of Checking Messages, and the target CTR Rate.
* Categorical columns (City, Preferred Messaging Channel, Type of Device, Type of CTA) are one-hot encoded using OneHotEncoder from sklearn to convert them into numerical arrays.

Below graphs show the comparison of Input values and the CTR Rate.

#### 

#### **Train-Test Split**

* The dataset is split into training and validation sets using train\_test\_split with a ratio of 80% for training and 20% for validation.
* A StandardScaler is used to scale numerical data to improve the performance of the neural network.

#### **Neural Network Model Creation**

* A neural network model is built using TensorFlow/Keras with the following architecture:

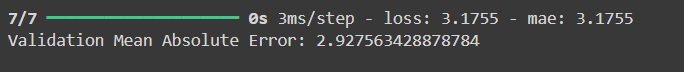
**Input Layer**: Accepts the transformed and scaled input data.

**Hidden Layers**:

* + - First Layer: 128 units, ReLU activation.
    - Second Layer: 64 units, ReLU activation.
    - Third Layer: 32 units, ReLU activation.

**Output Layer**: A single unit with linear activation for predicting the CTR rate.

* The model is compiled with the Adam optimizer and trained using Mean Absolute Error (MAE) as the loss function.



#### **Saving the Model**

* After training, the model is saved to a file (ctr\_model.h5) for later use in the Flask server.

### **B. Flask API Implementation**

#### **Flask Setup**

* The Flask web framework is used to create an API that accepts POST requests from the frontend.
* CORS (Cross-Origin Resource Sharing) is enabled using Flask-CORS to allow communication between the React frontend and Flask backend.

#### **Loading the Model and Preprocessors**

* The trained TensorFlow model (ctr\_model.h5) is loaded.
* The OneHotEncoder and StandardScaler fitted during training are also re-loaded to preprocess new incoming data from the API.

#### **Preprocessing Function**

* A function, preprocess\_input(), transforms incoming JSON data (city, time of day, etc.) into a format suitable for the model (one-hot encoding, scaling).

#### **API Route**

* The /predict\_ctr route receives POST requests containing user inputs.
* These inputs are preprocessed and passed into the model to generate a CTR prediction, which is returned as a JSON response.

### **C. React Frontend Implementation**

#### **Form Creation**

* A form in the React app collects inputs from the user:
  + **City**: Dropdown list of cities.
  + **Time of Day**: Dropdown list of times in 24-hour format.
  + **Preferred Channel**: Dropdown of communication channels (email, WhatsApp, SMS).
  + **Type of Device**: Dropdown for device type (Android, iPhone).
  + **Type of CTA**: Dropdown for different CTA types (book now, know more, etc.).

#### 

#### **Form Handling and Submission**

* State management is handled using useState() for form data and CTR prediction result.
* On form submission, the data is sent via a POST request to the Flask API.
* If the request is successful, the predicted CTR is displayed on the screen. If not, an error message is logged in the console.

#### **Loading Indicator**

* A loading indicator (Loading...) is displayed while waiting for the prediction result.

1. **RESULT:**

The images below show the final result of the output along with the CTR Predictions.

