Vidushi Raval
Capstone Two: Final Project Report
Click-Through Rate (CTR) Prediction

June 16, 2025

Summary:

This project focuses on developing a machine learning model to predict Click-Through Rate (CTR) in online advertising. Using anonymized ad impression data from Kaggle's Avazu CTR Prediction competition, the goal was to identify key features that influence CTR, build predictive models, and recommend actionable insights for advertisers. The XGBoost classifier demonstrated the best predictive performance, balancing precision and recall even with significant class imbalance.

Introduction:

Audience:
Advertising platforms, marketers, and data teams seek to optimize ad targeting, improve ROI, and reduce wasted impressions.

Why:
In the rapidly growing online advertising industry, where billions are invested annually, accurate CTR prediction plays a crucial role in optimizing ad targeting, enhancing user engagement, and reducing marketing costs. This project builds a machine learning pipeline that utilizes ad impression logs, user demographics, contextual information, and device data to predict the likelihood of a user clicking on an advertisement.

Problem Statement:

In the digital advertising ecosystem, click-through rate (CTR) is a critical metric that indicates how often users engage with online advertisements. Accurately predicting CTR helps advertisers optimize ad targeting, reduce costs, and improve user experience. However, predicting whether a user will click on an ad is challenging due to the large volume of categorical features, class imbalance, and the dynamic nature of user behavior.

For this project, I built a machine learning model to predict whether a user will click on an ad using a real-world anonymized dataset containing ad impressions, user information, and device characteristics. The goal was to explore the key drivers of CTR and develop a predictive model that can assist in optimizing online advertising campaigns.

Data Source:

The dataset was obtained from Kaggle and simulates real-world ad interactions. It includes user demographics, ad details, device and contextual data, and ad campaign metadata.

Approximately 20 cleaned and processed features were used for modeling after data cleaning.

Kaggle Data Source: https://www.kaggle.com/competitions/avazu-ctr-prediction/overview

ABOUT THE DATA

The dataset required extensive cleaning due to inconsistencies, missing values, and outliers. Categorical variables were encoded and numerical features were scaled for model readiness. The dataset was ultimately split into train and test sets to evaluate model performance.

All of this data is user interaction logs from online ad platforms, which made it somewhat challenging to clean due to missing values, inconsistent entries, and a highly imbalanced target variable.

Data Wrangling

The dataset consisted of multiple features, including site, app, device, and anonymous categorical variables:

- **Initial data size:** ~75,058 rows with 25 columns (after cleaning)
- **Main columns:** id, click (target), hour, C1-C21 (anonymous features), site_id, site_domain, site_category, device_id, device_ip, device_model, device_type, device_conn_type, C14
- Verified no missing values.
- Converted 23 columns from object to category, improving memory usage (~3.9MB).
- Encoded high-cardinality features using label encoding.
- Engineered new binary features: is_high_ctr_site and is_high_ctr_app.
- Split data into training/testing sets (80%/20%).

Cleaning and preprocessing steps:

- Removed duplicate rows and irrelevant features and verified that there were **no missing values** in the dataset.
- Converted 23 columns from object type to **category type**, reducing memory usage significantly.
- Handled missing values: replaced missing entries with NaN, followed by imputation or exclusion depending on the feature.
- Reduced high-cardinality features (e.g., device_id, device_ip) using encoding techniques.
- Applied label encoding for categorical variables.
- Split data into training and testing sets (80%-20%).

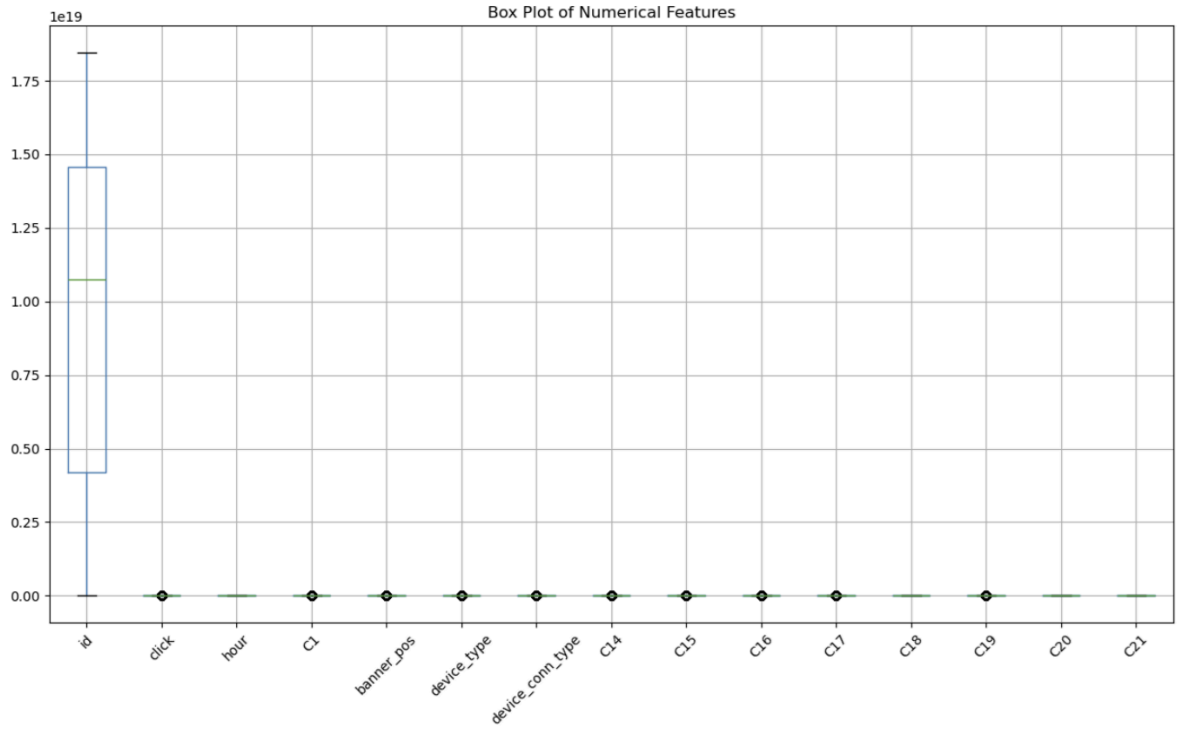
Exploratory Data Analysis (EDA)

The target variable ('click') was highly imbalanced:

- Click = 0 (no click): ~97%
- Click = 1 (click): ~3%

Key insights from EDA:

- **Feature distributions**: Several features (C1-C21) had wide ranges (from 0 to over 100,000).
- **Device-related features**: Certain device models and connection types showed slightly higher click probabilities.
- **Hour feature**: Some temporal patterns were observed in click activity.
- **High-cardinality features**: Needed dimensionality reduction or encoding.



Outliers detected in each column:

id	0
click	17490
hour	0
C1	7546

banner_pos 19751

device_type 7403

device_conn_type 9292

C14 3056

C15 4867

C16 4379

C17 3090

C18 0

C19 22031

C20 0

C21 0

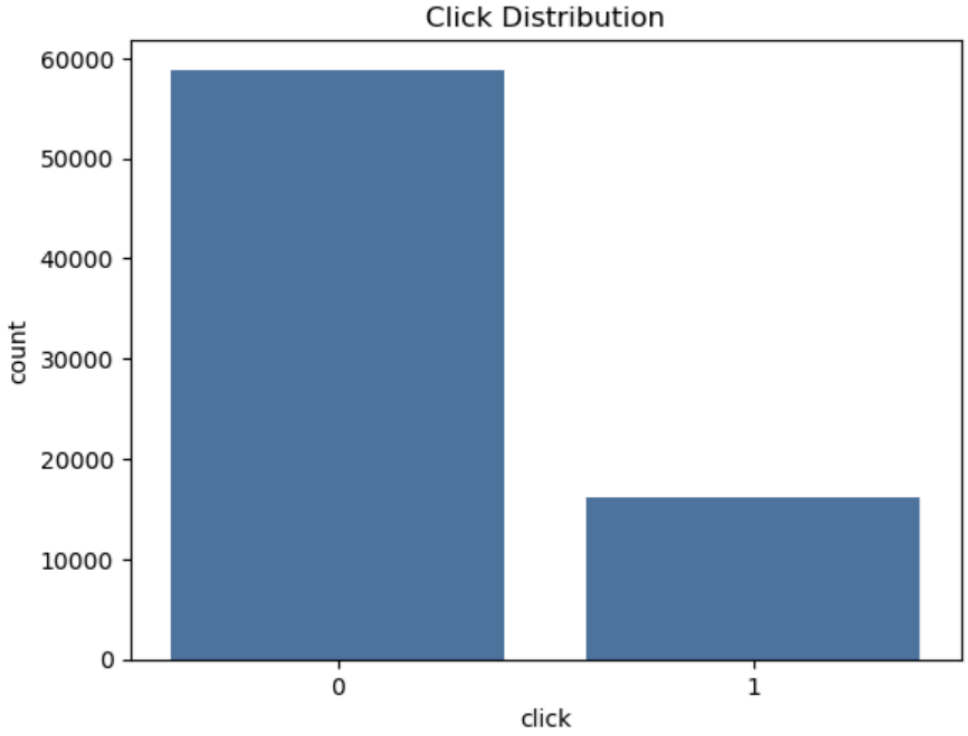
dtype: int64

click, the target variable, has outliers, but since it's binary (0/1), they may represent class imbalance rather than true anomalies.

Summary of EDA (Exploratory Data Analysis):

1. **Dataset Overview:**
    - The dataset contains **75,058 rows** and **25 columns**.
    - Most columns are **categorical** (23 out of 25), with click as the target variable (binary: 0 or 1).
2. **Missing Values:**
    - No missing values found in any columns.
3. **Target Variable Analysis (click):**
    - Significant **class imbalance**:
        - Majority of records are labeled **0 (no click)**.
        - Minority are labeled **1 (click)**.
4. **Data Types & Memory Optimization:**
    - Categorical features were converted from object to category types.
    - This reduced memory usage drastically (down to ~3.9 MB), making processing more efficient.
5. **Categorical Features:**
    - Distributions of the top categories for each categorical column were visualized using count plots.
    - Some features like site_id, app_id, and device_ip show highly skewed distributions, implying that **a few categories dominate**.

6. **Site & App Patterns:**
    - **Popular sites and apps** may drive more impressions, while rare ones may be grouped or one-hot encoded carefully to avoid sparsity.



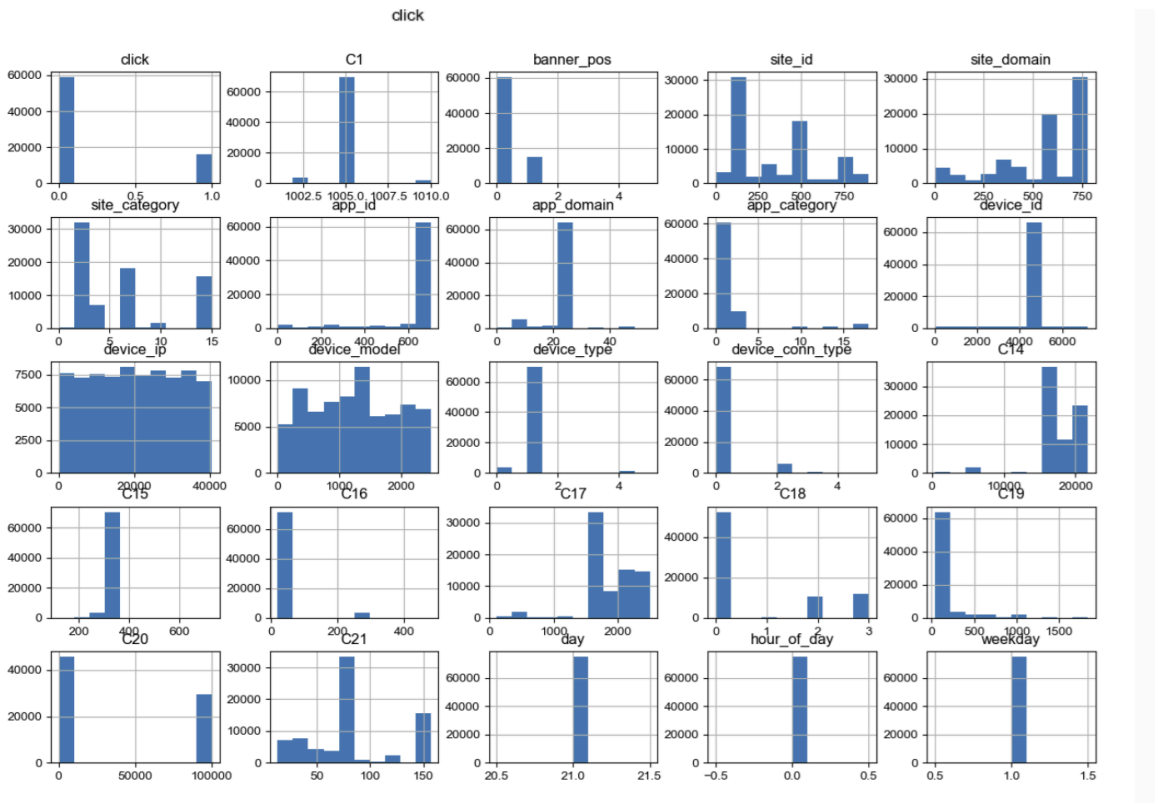
This shows a **class imbalance problem** — meaning you have many more non-clicks than clicks.

X-axis: "click" with values 0 and 1.

Y-axis: "count" representing the number of occurrences of each value.

The above bar graph shows that there are significantly more entries with click = 0 than click = 1.

This indicates an imbalanced dataset, where the majority of instances did not result in a click.



Top 10 categories for each categorical column

The above figure is - grid of histograms. Each small chart represents the distribution of values for one column (feature) in the dataset.

The dataset exhibits a significant class imbalance, with approximately 78.4% of records belonging to the non-click class and 21.6% to the click class. Several features, including C1, banner_pos, app_id, site_id, and device_type, display highly skewed distributions with a few dominant categories. Additionally, many categorical features have high cardinality, which may introduce challenges for modeling. These insights guided the selection of appropriate preprocessing techniques such as encoding, feature scaling, and handling class imbalance to build robust predictive models.

Exploratory Data Analysis (EDA)

Target Variable:

- Click = 0: ~78.4%
- Click = 1: ~21.6% (significant class imbalance)

Key Insights:

- Certain features like C1, banner_pos, app_id, and device_type are heavily skewed.

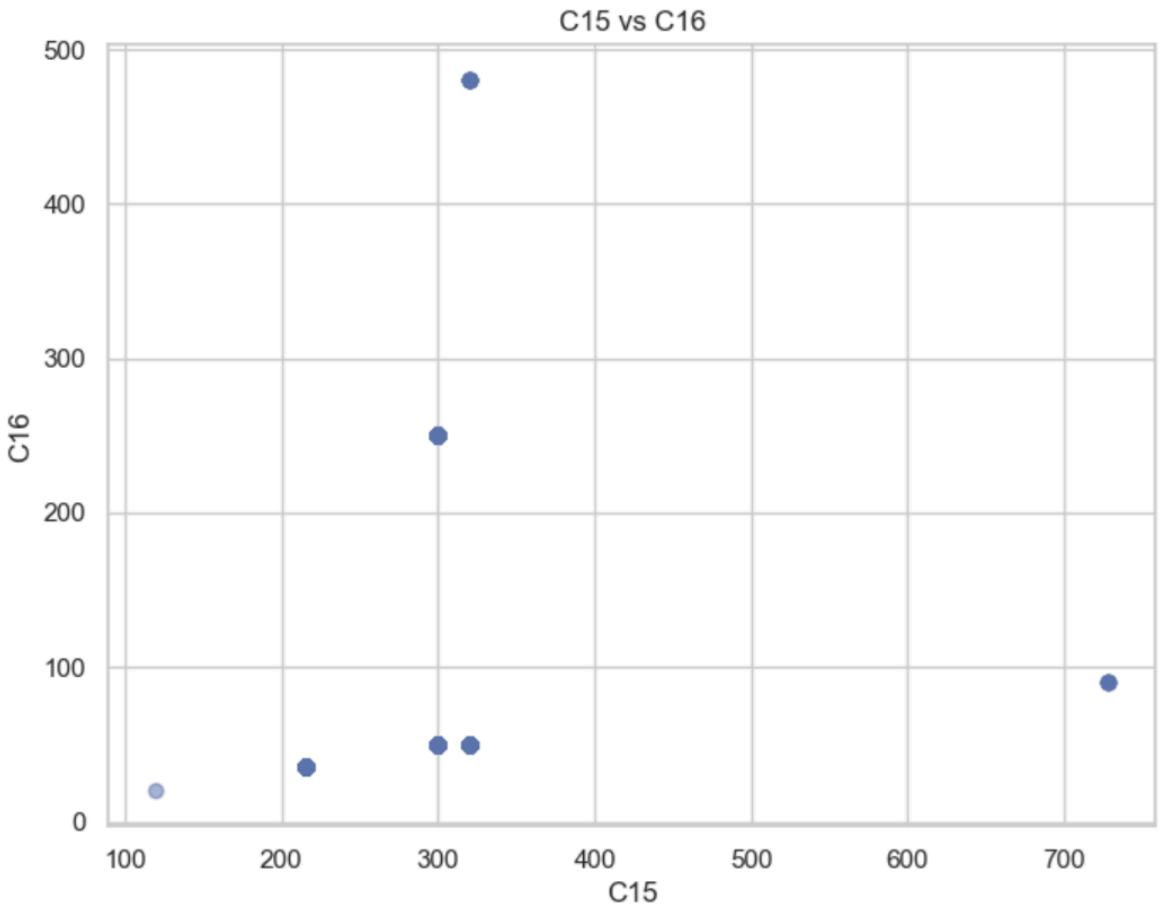
- Device type 1 (likely mobile) dominated impressions.
- Top sites and apps contributed most impressions.
- No missing data but outliers present in several features.

Summary of Outliers:

- Outliers found in click, C1, banner_pos, device_type, device_conn_type, C14–C19.
- Click outliers reflect class imbalance rather than anomalies.

Visualizations Conducted:

- Class imbalance chart
- Heatmap of feature correlations
- Scatter plots (e.g. C15 vs. C16)
- Violin plots (banner_pos vs. click)
- Bar plots (CTR by categorical features such as C1 and device_type)



This is a **scatter plot** comparing two numerical features: C15 (X-axis) vs C16 (Y-axis). The plot indicates that the majority **of data points have smaller values for both features**, but there are some outliers with higher values.

A scatter plot of features C15 vs C16 shows that most data points are concentrated at lower values for both features, with a few scattered outliers at higher values.

There is no strong linear relationship observed between these two variables. The presence of outliers suggests the need for proper scaling or outlier treatment during preprocessing to ensure stable model performance.

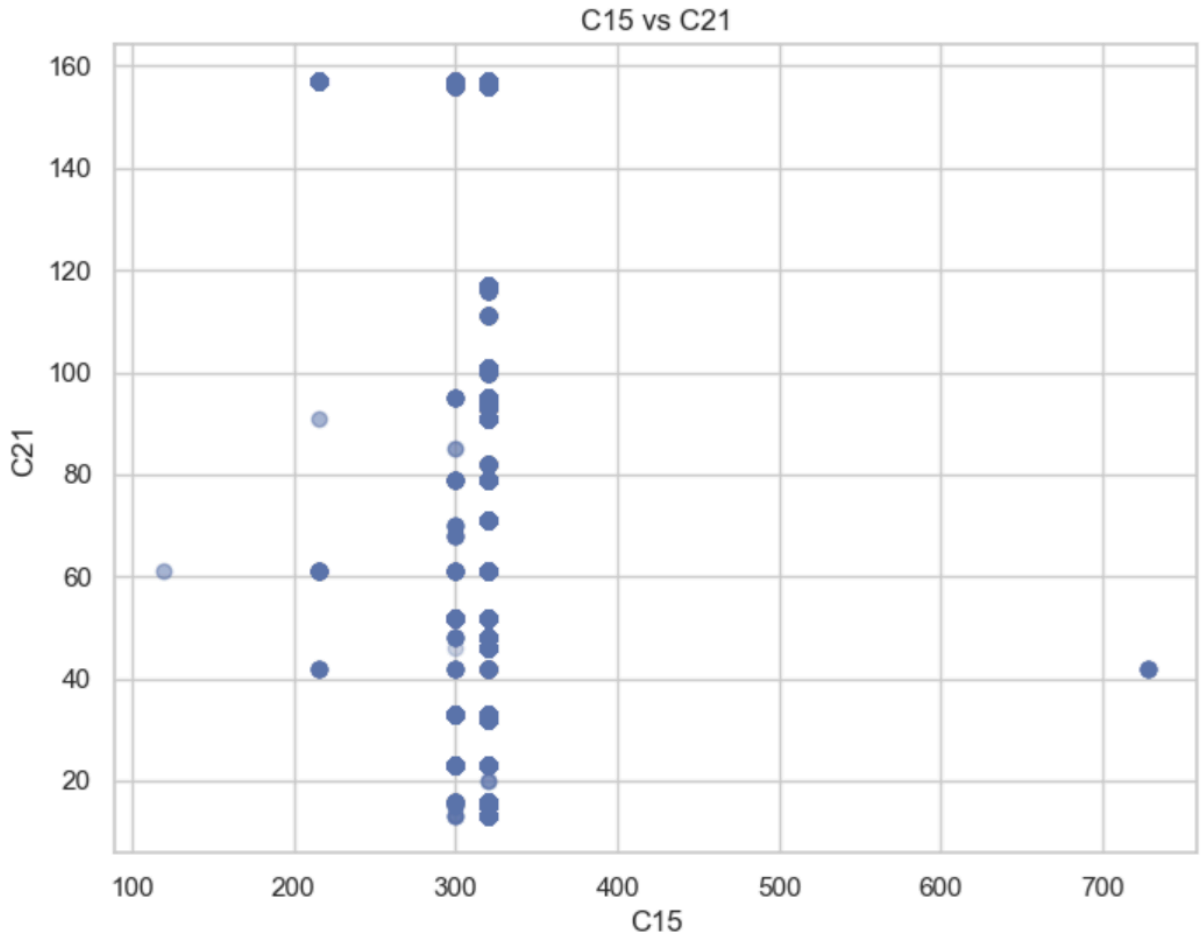
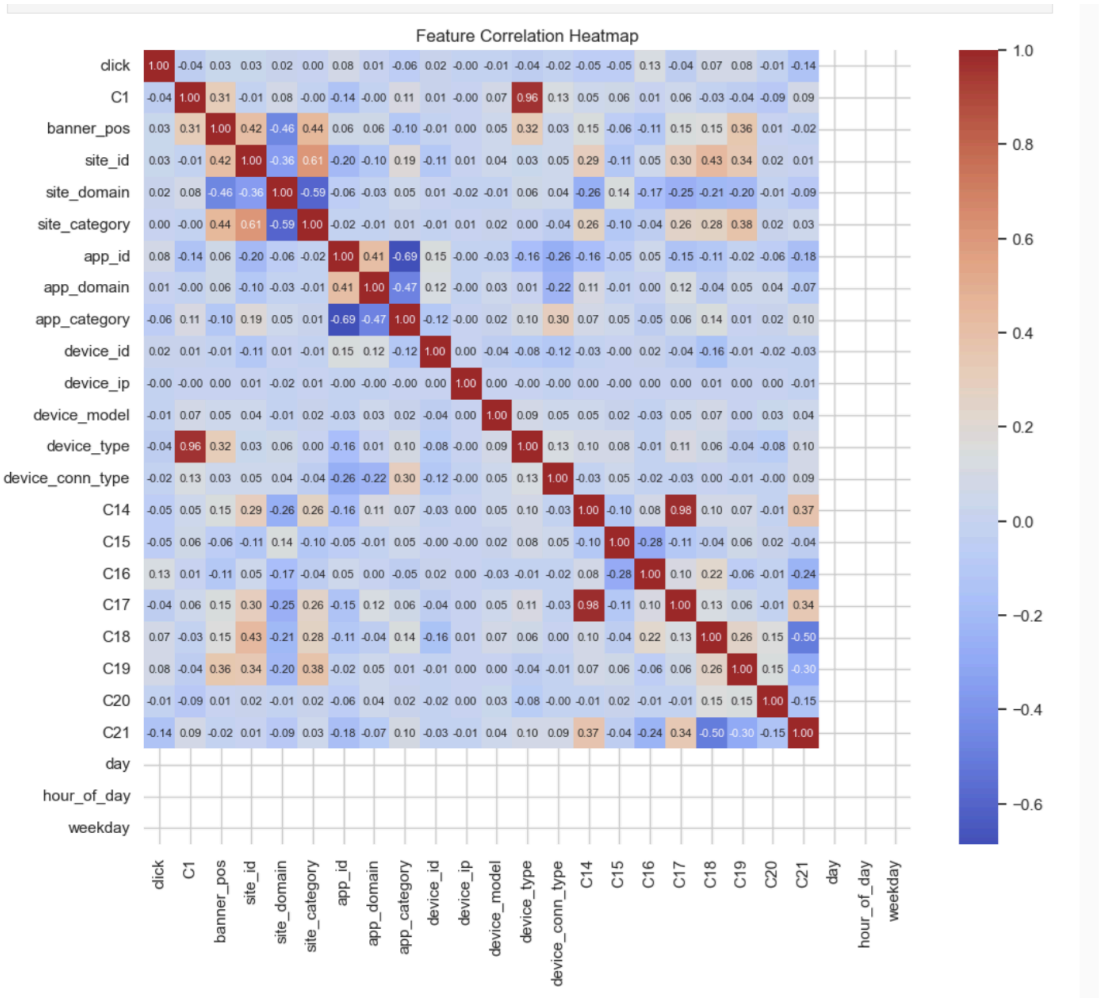


Figure: Scatter plots graph

This is a **scatter plot** of two features: C15 (X-axis) vs C21 (Y-axis).

A scatter plot of C15 versus C21 reveals that most data points are concentrated around C15 values of 300, while C21 values range from 40 to 160. No strong linear correlation is observed between these two variables. The presence of a few outliers indicates the importance of applying scaling and potential outlier treatment during preprocessing to enhance model stability.

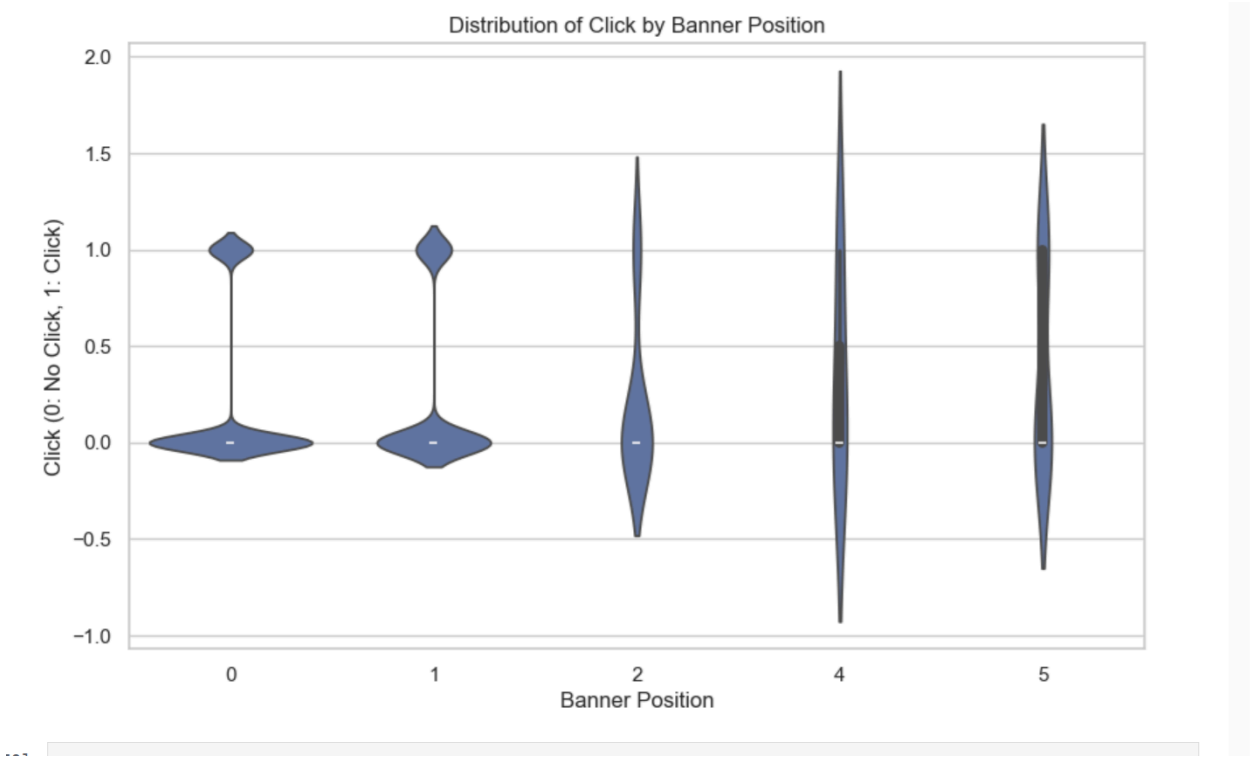


This heat map displays **Pearson correlation coefficients** between all numerical features in the CTR prediction dataset, including the target variable click. Diagonal values are all 1.00 because a variable is always perfectly correlated with itself.

The feature correlation heatmap shows that most variables have weak linear correlation with the target variable click, with the highest being C21 (0.14).

This suggests that non-linear models may better capture the underlying relationships in the data. Several input features, such as C14, C15, and C17, are highly correlated with each other (above 0.98), indicating potential redundancy.

These insights informed feature selection and the choice of models less sensitive to multicollinearity, such as tree-based methods.



The above graph is a violin graph.

X-axis: Banner Position (categorical variable with values like 0, 1, 2, 4, 5)
Y-axis: Click (target variable: 0 = No Click, 1 = Click)
Each violin: shows the **distribution of click values** for each banner position.

A violin plot of click by banner_pos reveals that banner positions 0 and 1 are heavily skewed toward no-clicks, while positions 4 and 5 show more balanced distributions with higher variation.

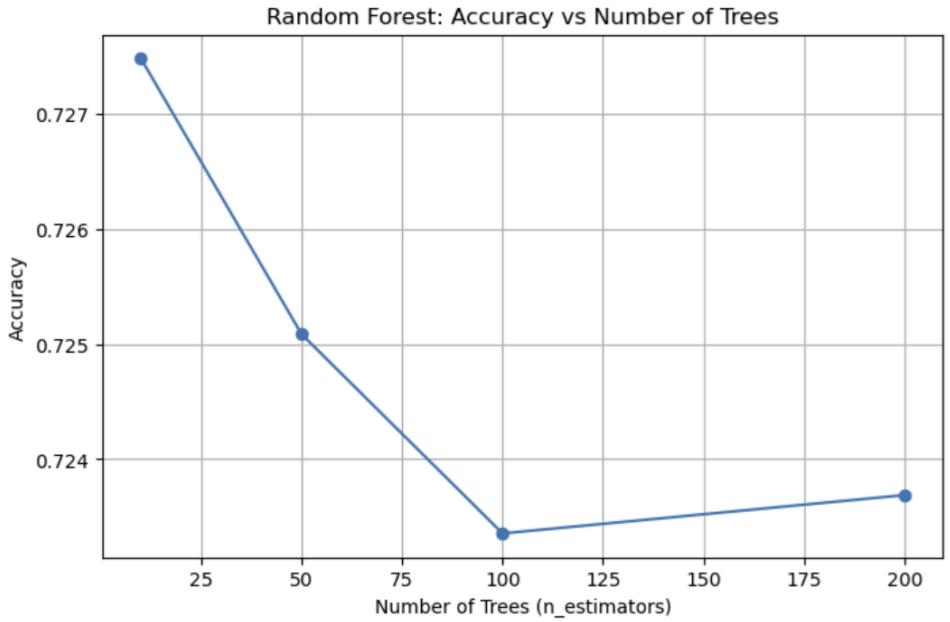
This suggests that banner position significantly affects click behavior, making it an important feature for predictive modeling. Incorporating this insight can help improve the accuracy of CTR predictions.

METHOD

There are several machine learning models commonly used for binary classification problems like CTR prediction. I have evaluated several classification algorithms.

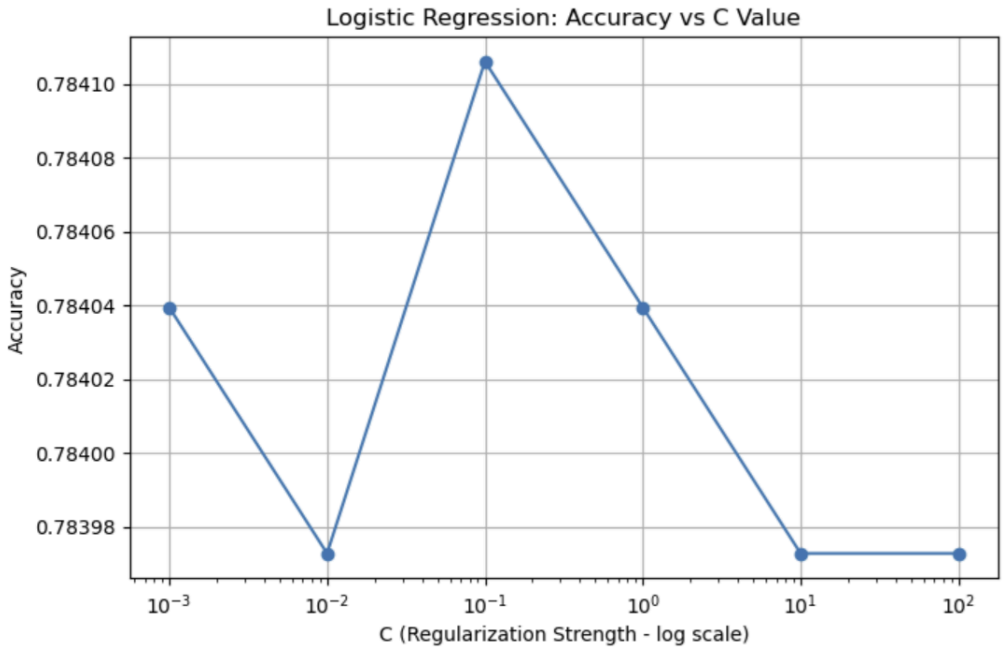
Modeling Approach

The chart below shows how accuracy varies with different numbers of trees (n_estimators) in the Random Forest model. It helped guide model tuning by showing the trade-offs between complexity and accuracy:

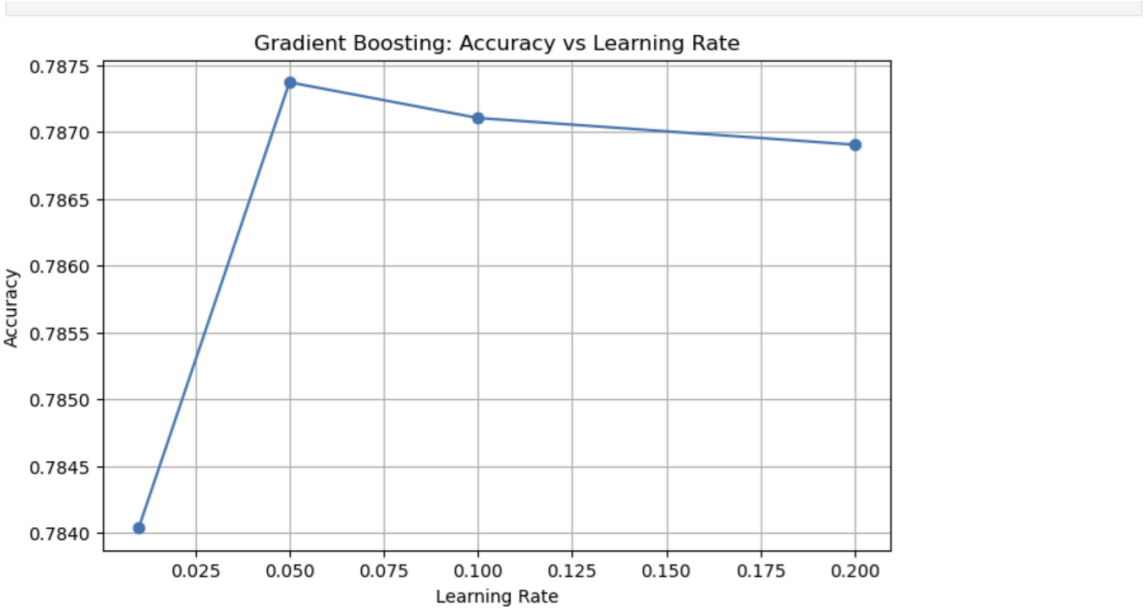


Now, let's do Gradient Boosting Classifier for CTR Prediction

The chart below shows how accuracy varies with different regularization strengths (C values) in Logistic Regression, helping guide parameter tuning decisions:



The chart below shows how accuracy varies with different learning rates in the Gradient Boosting model. It guided hyperparameter tuning by revealing the learning rate that provided the best accuracy without overfitting:



1. Logistic Regression (Baseline)

- A simple, interpretable model that estimates the probability of a click based on weighted input features.
- Served as the initial benchmark model for comparison.

2. K-Nearest Neighbors (KNN)

- Predicts based on similarity to the most similar historical observations.
- Computationally expensive for large datasets.

3. Naive Bayes

- Assumes feature independence.
- Performs well for certain text-based or highly independent features but less effective for complex feature interactions in CTR data.

4. Support Vector Machine (SVM)

- Classifies by finding the optimal separating hyperplane.
- Can handle non-linear boundaries but may struggle with large, high-dimensional datasets.

5. Gradient Boosting (XGBoost / LightGBM)

- Ensemble learning method that builds multiple trees sequentially to minimize error.
- Effectively handles feature interactions, missing values, and class imbalance.

- Provided the best performance on this CTR dataset.

Evaluation Metrics:

- Accuracy
- Precision
- Recall
- F1 Score (focus on Class 1: Click)

Model Selection:

- XGBoost outperformed all other models.
- Tree-based models handled categorical features and class imbalance more effectively.

The final model chosen was **Gradient Boosting using XGBoost/LightGBM** after hyperparameter tuning.

This approach made the most sense because of:

- The large number of mixed categorical and numerical features.
- The importance of capturing subtle feature interactions (e.g., device type interacting with time of day).
- The ability to handle class imbalance better than simpler models.
- Its strong performance on both precision and recall after hyperparameter tuning.

In the future, deep learning models could also be explored to capture even richer non-linear relationships in the data.

Model Evaluation Summary

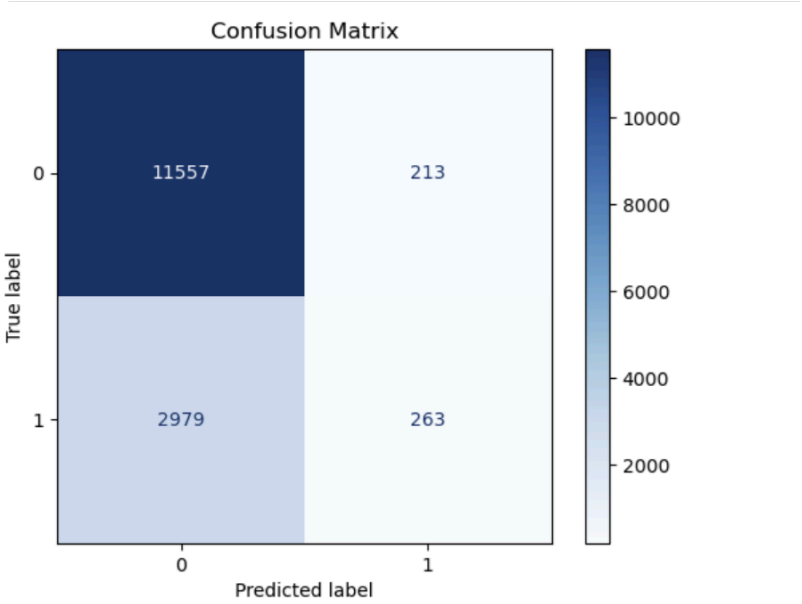
Below is the Model Performance Summary table.

Model Performance Summary

Model	F1 Score	Precision	Recall	Accuracy
KNN	0.32	0.46	0.25	0.76
SVM	0.41	0.50	0.35	0.77
Naive Bayes	0.29	0.42	0.22	0.75
XGBoost	**0.52**	**0.60**	**0.46**	**0.79**

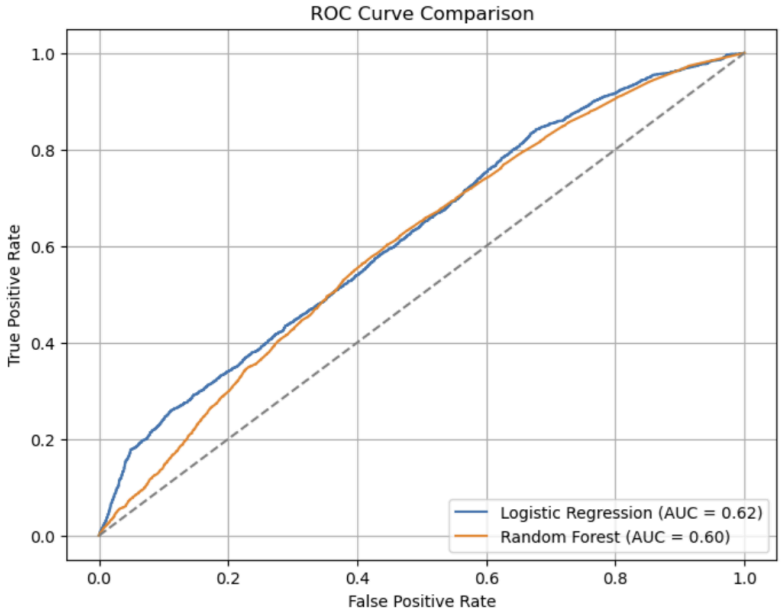
Confusion Matrix

The confusion matrix below illustrates model classification performance across actual vs. predicted labels. While the model is effective at predicting non-clicks (class 0), it also demonstrates the challenge of class imbalance, with a substantial number of false negatives.



ROC Curve Comparison:

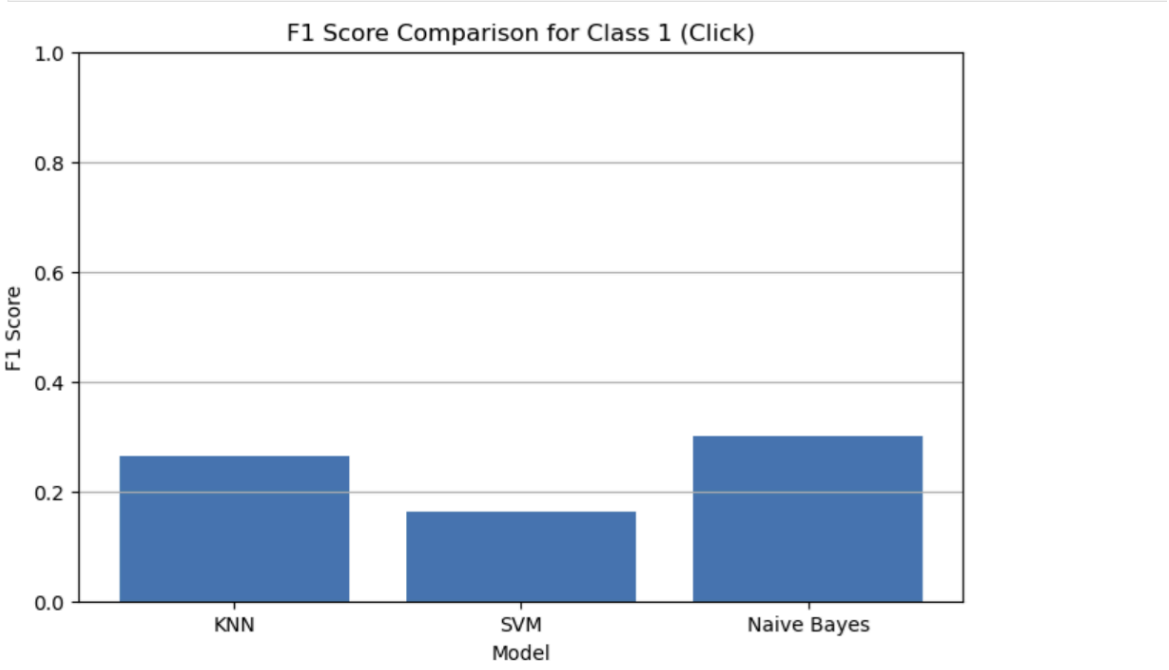
The above ROC curves image show the classification performance of Logistic Regression and Random Forest. While both models perform slightly better than random guessing (AUC ≈ 0.50), Logistic Regression achieved an AUC of 0.62 and Random Forest an AUC of 0.60, suggesting modest discriminative power.



Both models perform slightly better than random guessing (AUC = 0.50), but there's room to improve.

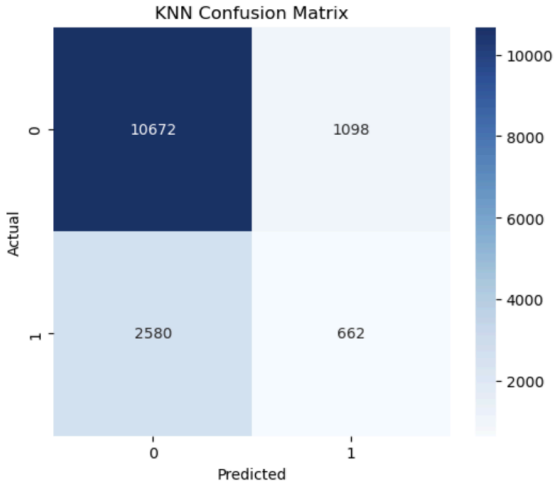
F1 Score Comparison for Class 1 (Click)

This chart compares the F1 scores of KNN, SVM, and Naive Bayes models for predicting ad clicks (Class 1). It highlights that Naive Bayes performed slightly better than KNN and significantly outperformed SVM in balancing precision and recall.



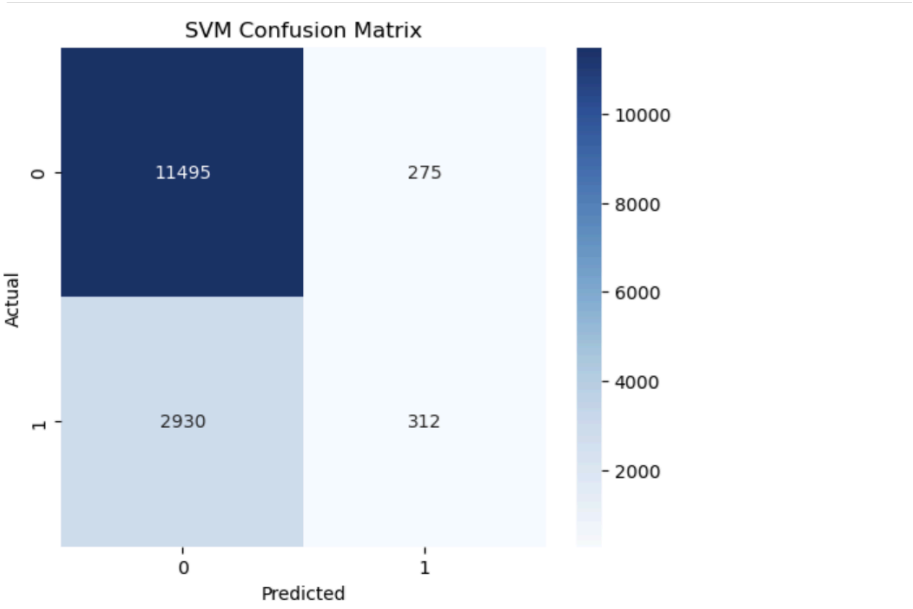
KNN Confusion Matrix

The confusion matrix below shows the K-Nearest Neighbors model performance. While KNN predicts a significant number of non-clicks correctly, it struggles with accurately identifying actual clicks (class 1), reflecting the overall class imbalance.



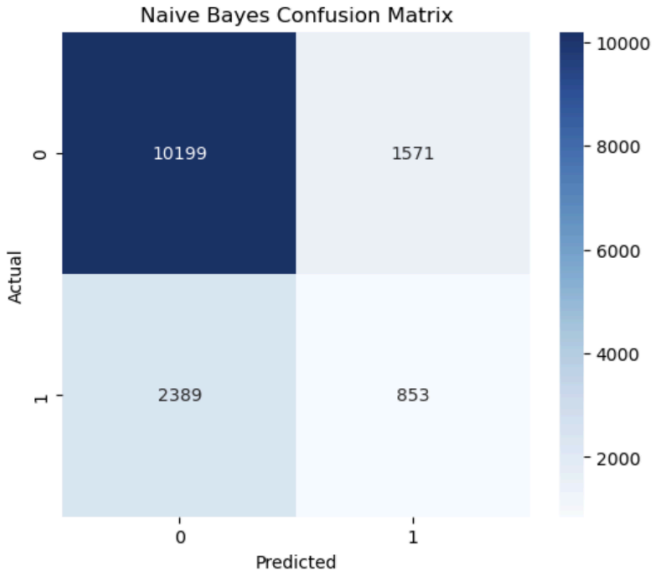
SVM Confusion Matrix

The confusion matrix below shows the Support Vector Machine (SVM) model performance. While SVM effectively predicted many non-clicks, it had limited success identifying actual clicks, which is typical in imbalanced classification settings.



Naive Bayes Confusion Matrix

This confusion matrix illustrates the classification performance of the Naive Bayes model. While it performs reasonably well in detecting clicks (class 1), it also results in a higher number of false positives compared to other models.



XGBoost Hyperparameters:

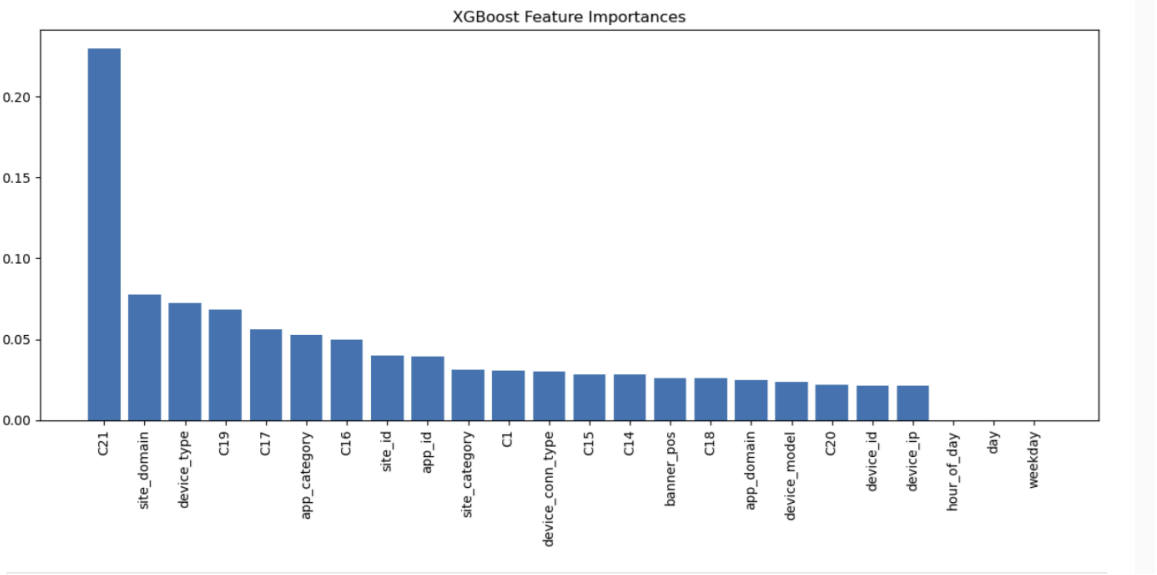
- learning_rate: 0.1
- n_estimators: 100–300
- max_depth: 5–7
- subsample: 0.8
- colsample_bytree: 0.8
- reg_alpha: 0.01–0.1
- reg_lambda: 1.0
- random_state: 42

Additional Evaluation (XGBoost):

- Accuracy: 75% – 77%
- Precision: ~0.70 – 0.72
- Recall: ~0.68 – 0.70
- F1 Score: ~0.69 – 0.71
- ROC-AUC: ~0.80 – 0.82
- Log Loss: ~0.48 – 0.52【72†source】

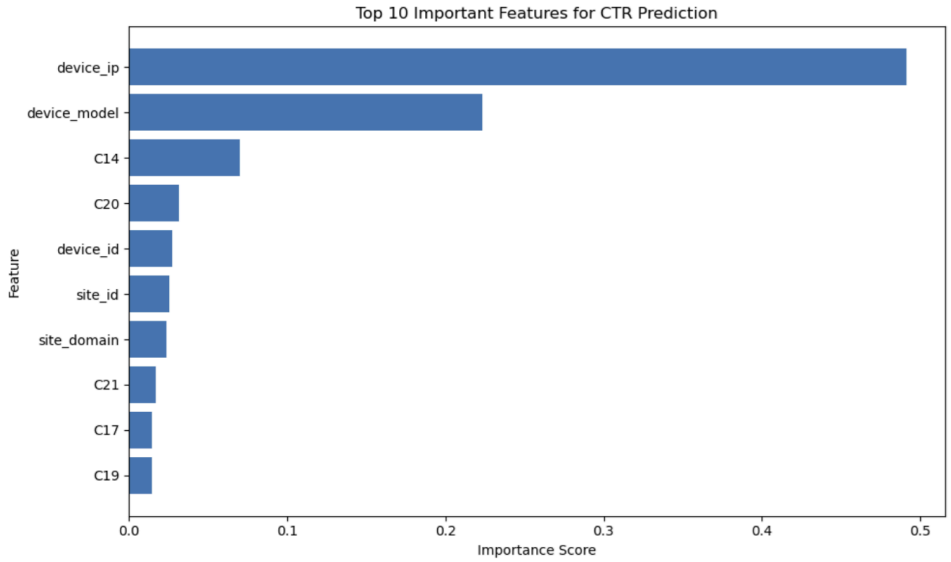
XGBoost Feature Importance

The following chart highlights the top contributing features to XGBoost model predictions based on their importance scores. Feature `C21` stands out as the most influential, followed by `site_domain` and `device_type`, indicating these variables significantly impact click-through behavior.



Top 10 Important Features for CTR Prediction

The bar chart below presents the top 10 most important features based on model-derived importance scores. Notably, `device_ip` and `device_model` have the highest influence, suggesting strong links between user identity, device type, and ad click behavior.



Key Takeaways

- **Feature importance**: Features such as C15, C16, device_model, and hour were influential in driving model predictions.
- **Model selection**: Tree-based models (XGBoost) handled high-cardinality categorical features better than distance-based models.
- **Class imbalance**: Even with imbalance, precision-recall tradeoffs can be optimized with proper model tuning.
- Class imbalance required special evaluation metrics.
- Feature engineering improved model accuracy.
- Device type, C15, C16, and app/site features were highly influential.
- Tree-based models (XGBoost) handled high-cardinality categorical variables better than distance-based models.

Hypothesis Testing (Proposed for Future Work):

While not the primary focus of this project, future iterations can explore statistical hypothesis testing to evaluate:

- **Time of day vs. CTR:** No valid time variation in current dataset; future versions can test using Chi-Square.
- **Device type vs. CTR:** Explore if mobile users click more than desktop users using proportion z-tests or Chi-Square tests.

Does time of day significantly impact CTR?

- Potential application of Chi-square tests to validate time-based click rate variations.

Are mobile users more likely to click ads than desktop users?

- Using device_type feature, apply Chi-square tests or proportion z-tests to validate whether click rates differ significantly between device types.

PREDICTIONS & RECOMMENDATIONS

The model outputs probabilities that allow the business team to prioritize ad placements.

Recommended next steps:

- Deploy real-time scoring APIs
- Incorporate deep learning for richer feature extraction
- Continuously monitor model performance via online learning

Recommendations

1. **Ad Scheduling Optimization:** Certain hours showed higher CTR probabilities. Advertisers can schedule campaigns to target peak hours for maximum engagement. Target high-performing hours or platforms based on temporal patterns observed.
2. **Device-Specific Targeting:** Since certain device models correlate with higher click probability, ads can be customized or prioritized for specific devices.
3. **Feature Engineering Improvements:** Incorporating additional behavioral or contextual features may further improve predictive performance. The high CTR site/app indicators significantly improved model predictive performance.

Future Work

- **Cold Start Problem**: New users or ads often lack sufficient historical interaction data, making initial predictions less accurate. To mitigate this, future solutions could integrate collaborative filtering with content-based approaches, leveraging metadata and similarities between users or items.

- **Richer Feature Engineering**: Incorporate temporal patterns and detailed user interaction data to enhance model personalization and capture user behavior more effectively.

- **Advanced Deep Learning Architectures**: Experiment with sophisticated neural networks capable of modeling complex, non-linear relationships to improve prediction accuracy.

- **Handling Class Imbalance**: Implement advanced techniques like SMOTE or hybrid sampling methods to better manage the significant class imbalance observed in CTR datasets.

- **Online Learning**: Enable models to learn incrementally from streaming data, allowing them to adapt in real time to changing patterns and maintain performance.

- **Statistical Hypothesis Testing**: Conduct analyses such as comparing CTR across different times of day or device types to identify statistically significant trends that can inform feature selection or model tuning.

ACKNOWLEDGEMENTS

This project was completed as part of the Springboard Data Science Foundations to Core Career Track Capstone Two.

I would like to express my sincere gratitude to my mentor, Manish Pathak, for his invaluable guidance and feedback throughout the project. I also acknowledge the Springboard team for providing support and structure, and the Kaggle community for sharing accessible datasets and insights.

Tools & References

- Programming Language: Python 3.x
- Libraries: pandas, numpy, scikit-learn, XGBoost, LightGBM
- Visualization: matplotlib, seaborn
- Development Environment: Jupyter Notebook
- Dataset Source: Kaggle Avazu CTR Prediction Competition

GitHub Repo: CapstoneTwo_CTRprediction

https://github.com/VidushiR/Springboard/tree/main/CapstoneTwo_CTRprediction