Trustworthy Hackathon

Lucas, Chloe, Daivya, Eric

Our Mission

In today's digital marketing landscape, the ability to precisely target advertisements is essential for maximizing user engagement and conversion rates. Our mission is to discovering useful information from data, leverage machine learning to analyze an extensive dataset, develop predictive models, and generate synthetic data to improve the accuracy and efficiency of ad targeting.

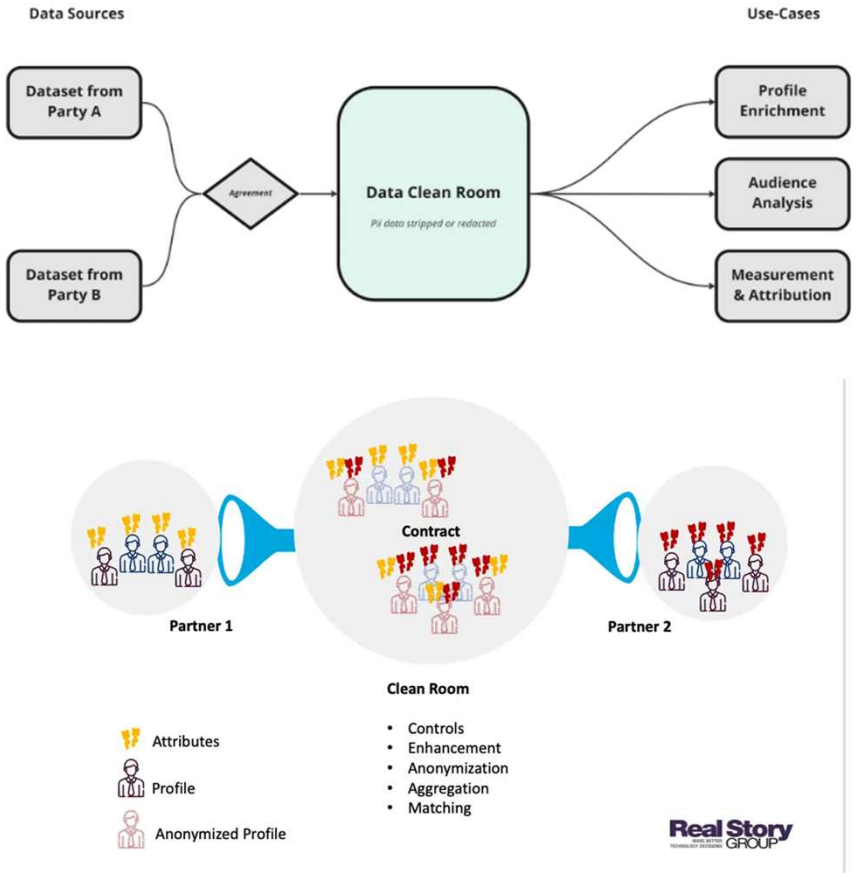
Part 1:
Data Security

Protecting Data in Use

Confidential Computing - Currently, there is a vast array of methods to protect data in storage or at rest, but there remains a gap when it comes to protect data in use. Our goal is to navigate an application of confidential computing in a multi-party data sharing scenario.

Goals - Protect data in use by ensuring the confidentiality and integrity of the code and the environment

Purpose - To provide a secure and confidential data sharing platform. In this case, the platform will allow for publishers and advertisers to confidentially combine their data together to extract machine learning insights without having their customer data leaked to the other party or accessed by malicious actors

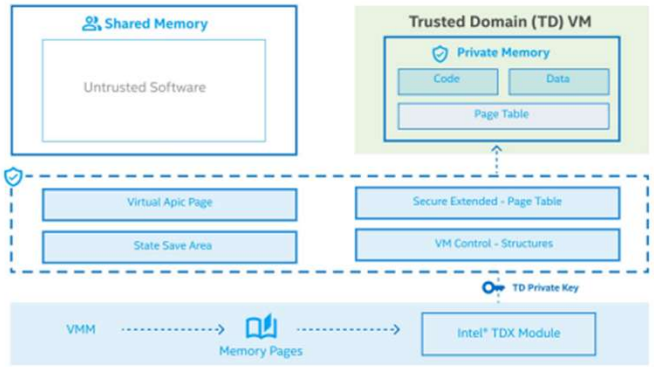


Technical Analysis

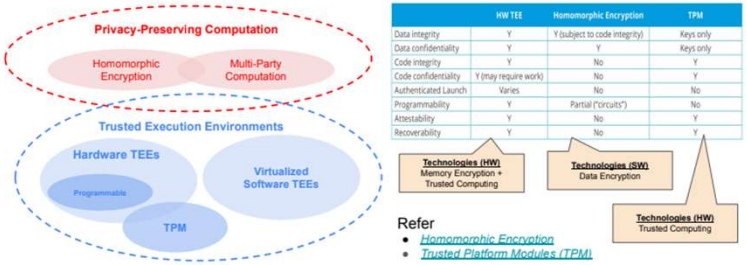
Resources/Technologies

- **Azure Confidential VM -** Provides a secure Ubuntu environment to run operations. Azure Confidential VMs provide numerous security benefits such as Intel TDX and TPM capabilities, isolation and data confidentiality. Also Azure products allow for immense scalability.
- **tpm2-tools library -** Allows for attestation utilizing the TPM. Can create endorsement keys to attest for the TPM itself and an attestation key to attest for the evidence. Also provides tools aiding with the modification and evaluation of PCR values and quote creation and verification
- **Azure Key Vault -** Provides a safe remote server to store keys. Can securely transport these keys through the Amazon CLI as they utilize TLS.
- **Azure Storage Containers -** Allows easy and secure transfer of files between confidential VMs. Easily scalable storage

Implementation - Please see the markup file in this GitHub Repository for an in depth explanation of the implementation described above
https://github.com/ChloeHouvardas/CC-TPM-Attestation/blob/main/instructionsRoundTwo.md

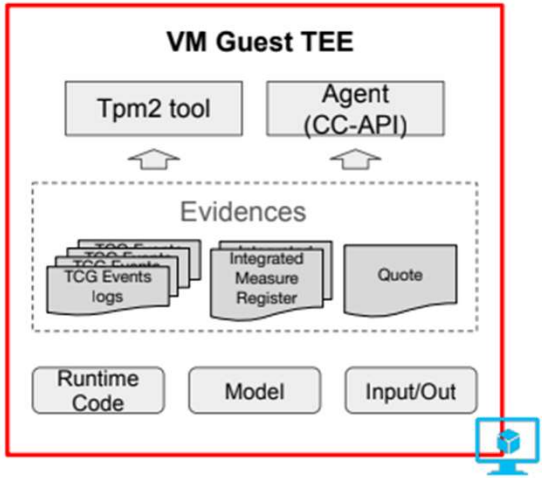


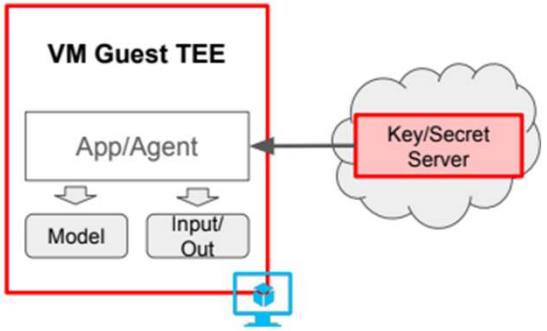
Intel TDX Utilized by Azure's Confidential VMs



Technical Implementation Steps

1) Setup Azure Confidential VMs (one for TEE one for remote attestation server
2) Download tpm2-tools on both VM
3) Send Encrypted Resources to TEE (via scp)
4) Create endorsement key (verifies root of trust) and use it to make attestation key
5) Collect evidence
    a) Calculate hash for code and data
    b) Entend to PCR
    c) Create quote
    d) Send quote to remote attestation server VM using azure storage
6) Verify quote
    a) Switch to attestation server VM
    b) Download quote from azure storage
    c) checkquote
    d) If valid, send decryption key to TEE
7) Decrypt data, run models
    a) Use the decryption key and the decrypt.py script to decrypt data
    b) Run models on data
    c) Output insights and synthetic data
    d) Encrypt data

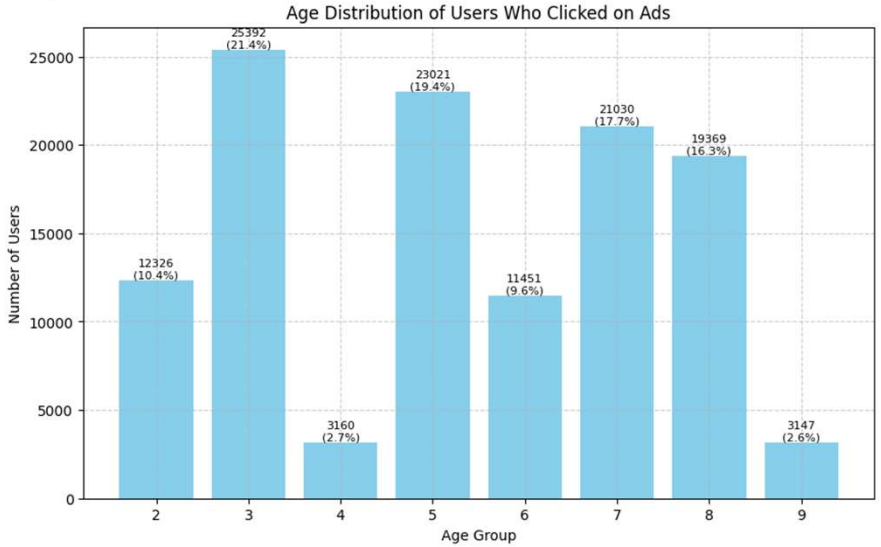


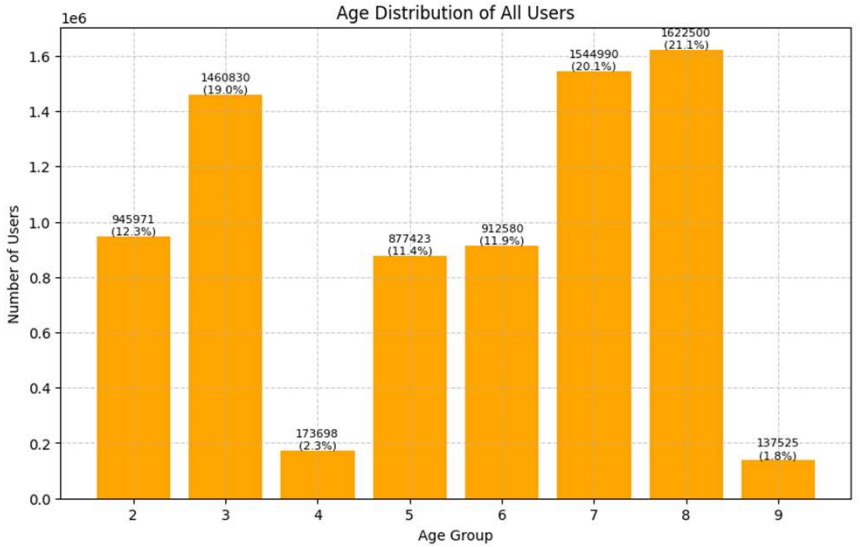


Part 2 Task 1: Data visualization

Data Visualisations

Age Group Distribution



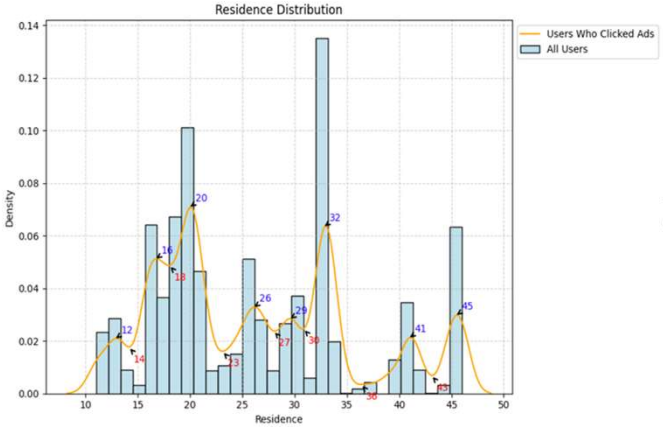


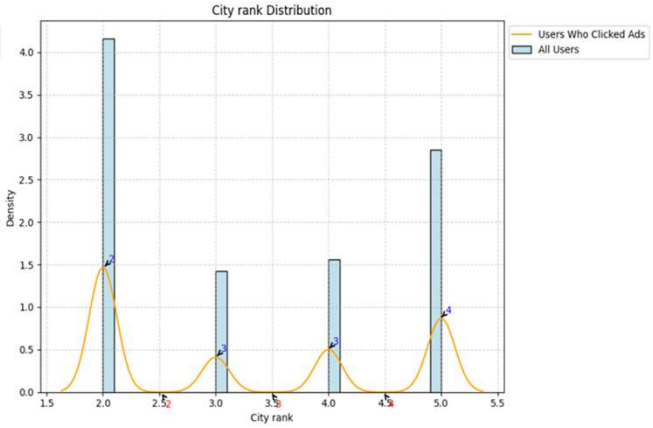
- The age group distribution for users who clicked on ads shows that the most responsive groups are Age Group 3 (21.4%), Age Group 5 (19.4%), Age Group 7 (17.7%), and Age Group 8 (16.3%).

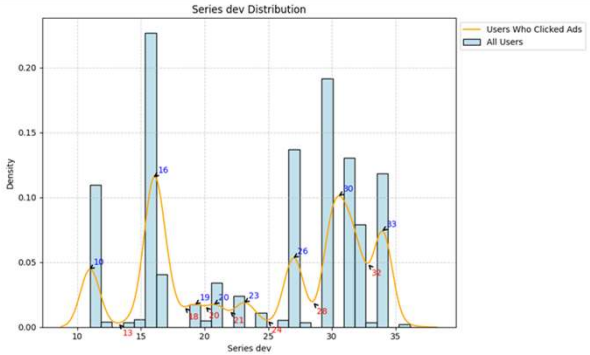
- This suggests that advertisements should be primarily targeted towards these age groups to maximize engagement. By focusing on the most responsive age groups, ad agencies can enhance the effectiveness of their campaigns and achieve better conversion rates.

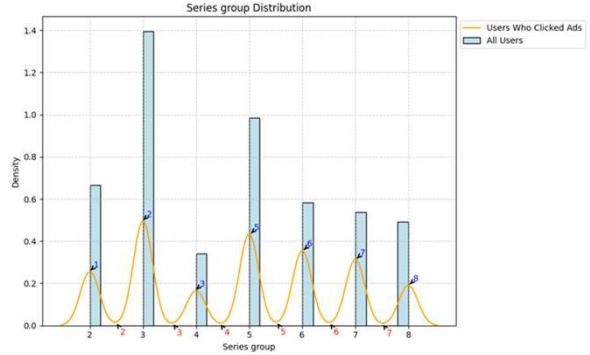
Data Visualisations

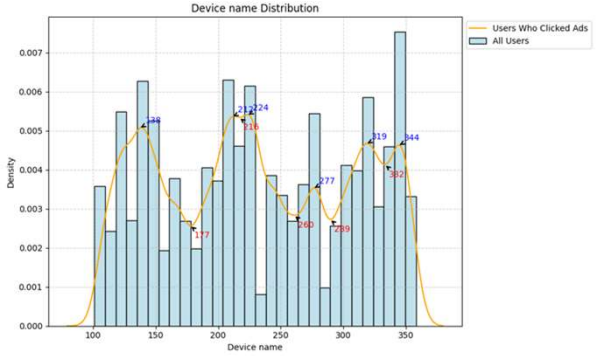
Geographic Distribution and Device Usage







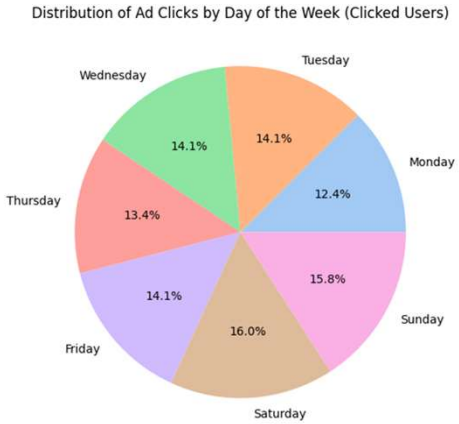


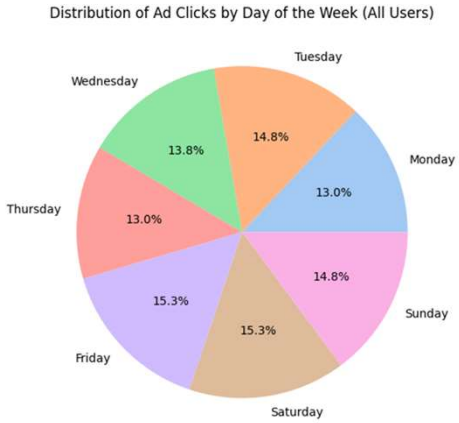


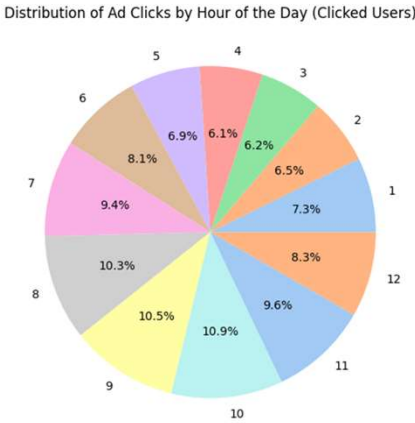
- The geographic distribution analysis reveals that users in residences around 20 and 32, city 319, and city ranks 2 show higher ad engagement compared to the overall user base.
- Users who click on ads prefer specific device characteristics: series_dev IDs 16 and 30, series_group ID 2, emui_dev ID 20, device_name IDs 138, 224, 344, device_size 2113 and 2405, and net_type 7. Targeting these characteristics can enhance ad effectiveness.

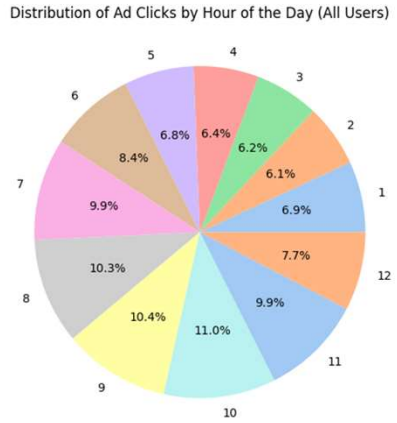
Data Visualisations

Engagement Patterns







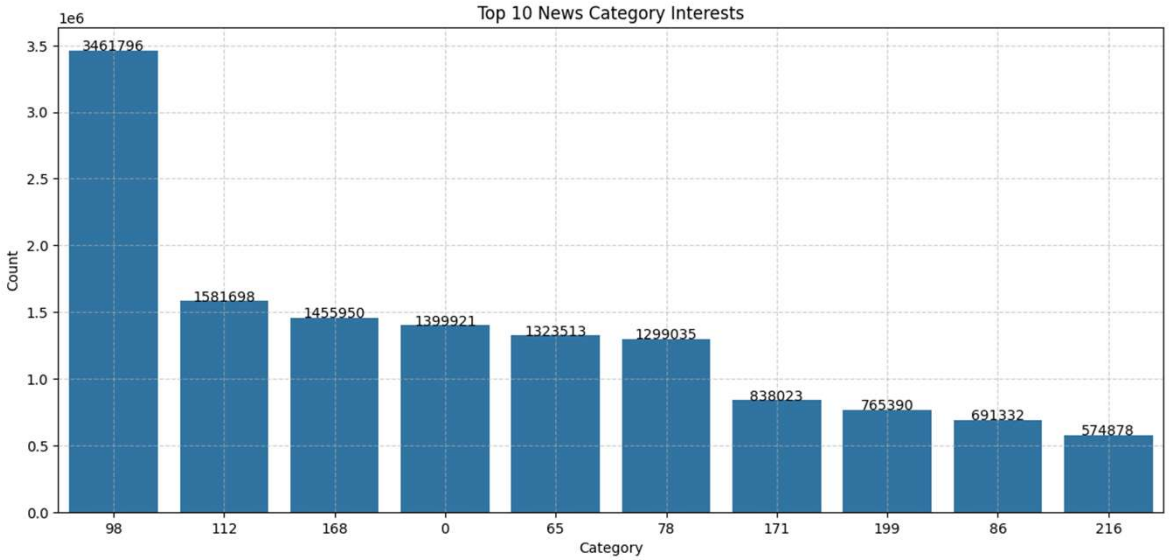


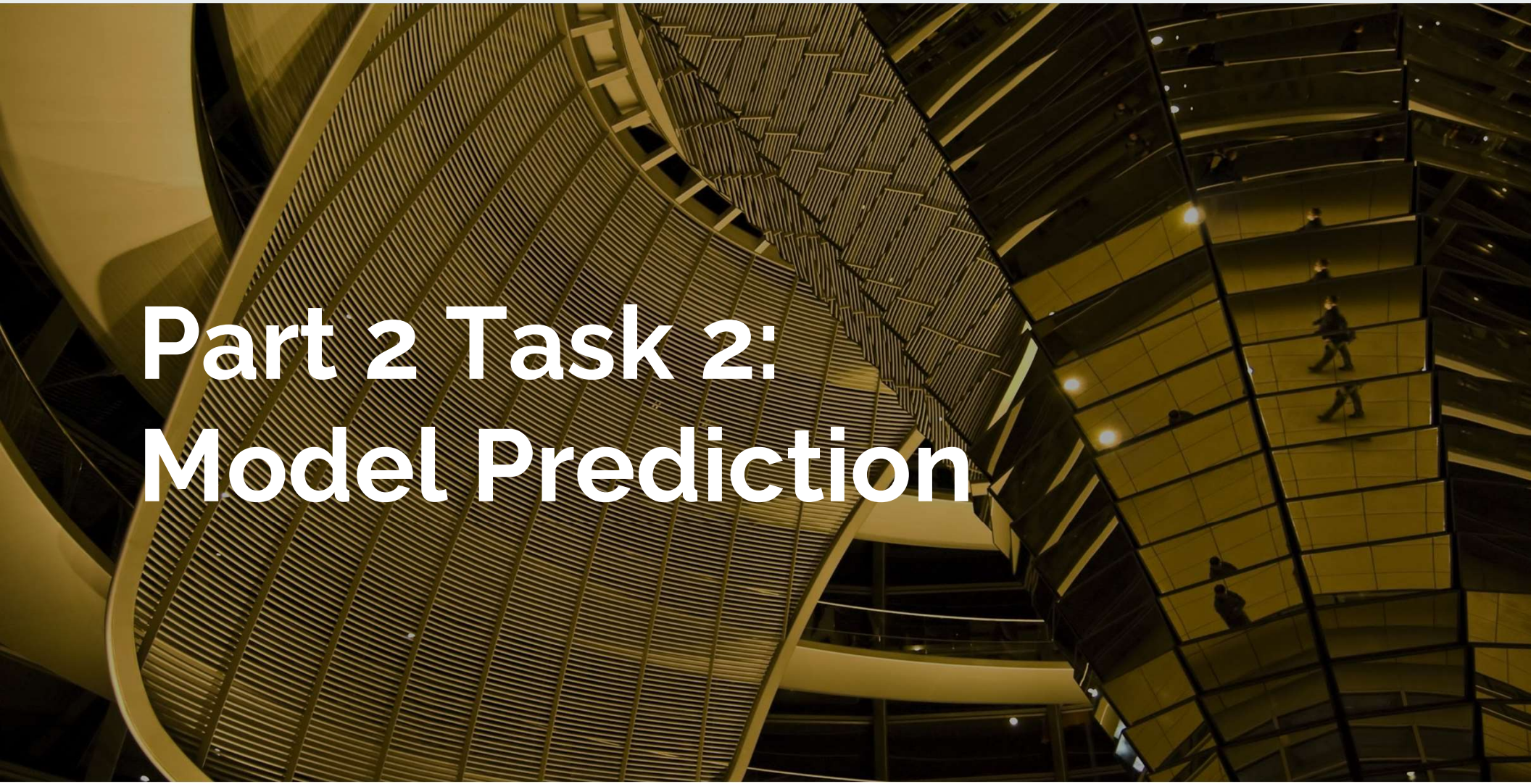
- Users click ads most on weekends, peaking at 16.0% on Saturday and 15.8% on Sunday. Weekday activity is balanced, with Monday at the lowest (12.4%). Ad clicks peak between 8 PM and 12 AM, especially at 10 PM (10.9%) and 11 PM (10.4%).
- These insights suggest that ad campaigns should be scheduled to maximize visibility during these high-engagement periods, particularly on weekends and during evenings to nights, to optimize user engagement and ad performance.

Data Visualisations

Content Preferences

- Category 98 is the most popular with 3.46 million users, followed by categories 112 and 168 with 1.58 million and 1.46 million users, respectively.
- Aligning ad content with these popular categories can increase click-through rates and user engagement.





Part 2 Task 2: Model Prediction

Data Cleaning

Featured_dataset.ipynb : this file clean the data, endow features to the data and create 2 csv file

Predict.ipynb: This file performs the data slicing, model training and provides answer to your question

Merging - Merging ads and feeds by variable user_id.

Variable featuring - give feature “clicked” to determine if the user interacted with the ads.

Variable Dropping - Dropping many variables from the dataset, they are either very biassed; they are duplicated; they are identifier(ids); they have too many unique value. (timestamp, city and residence)

Column Dividing - Splitting the column into multiple columns(like 23151^231^214^123, I separate this long list to multiple column. I think the numerical number could represent certain category, and after ^ is the subset of this category)

Some conclusion from the dataset

*This model only runs in a portion of the data

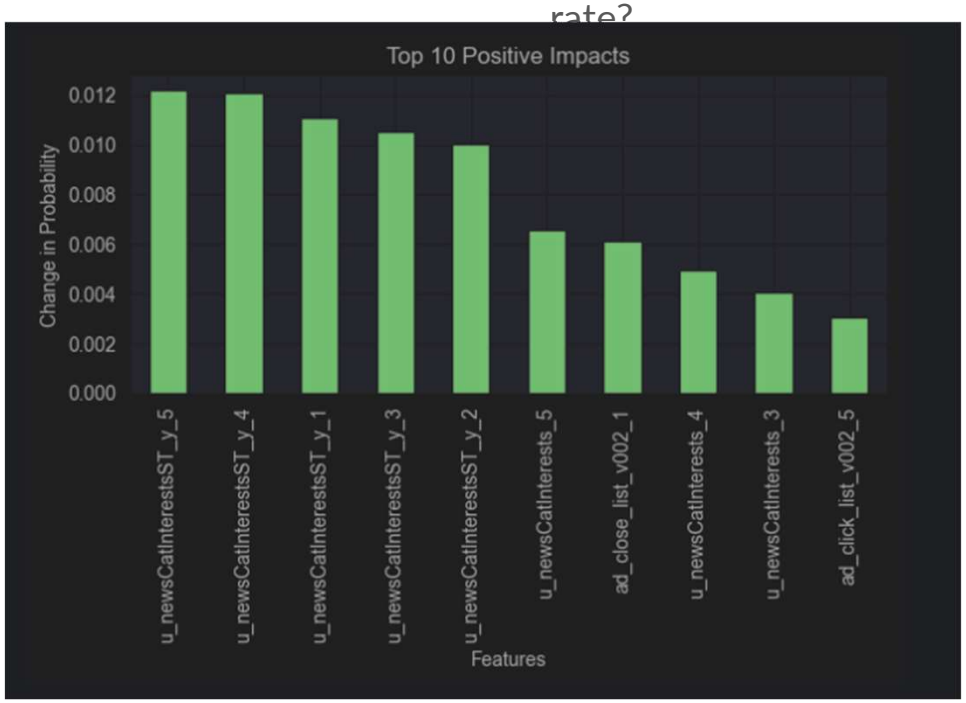
1 row | 1 rows × 5 columns pd.DataFrame

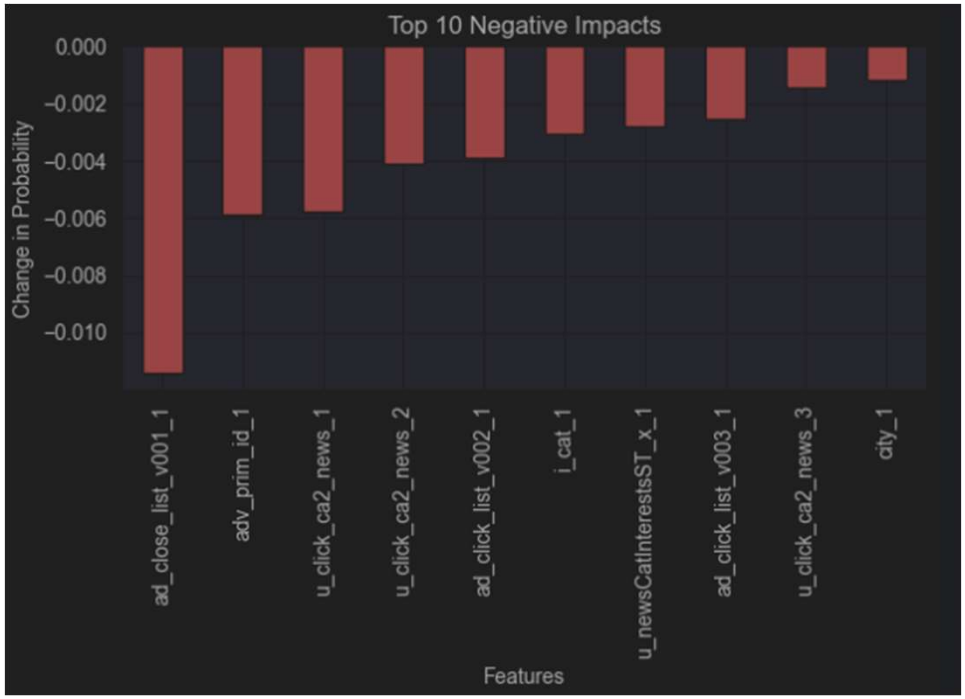
	method	training mse	training R2	test MSE	Test R2
0	random forest	0.00238	0.975626	0.00229	0.976551

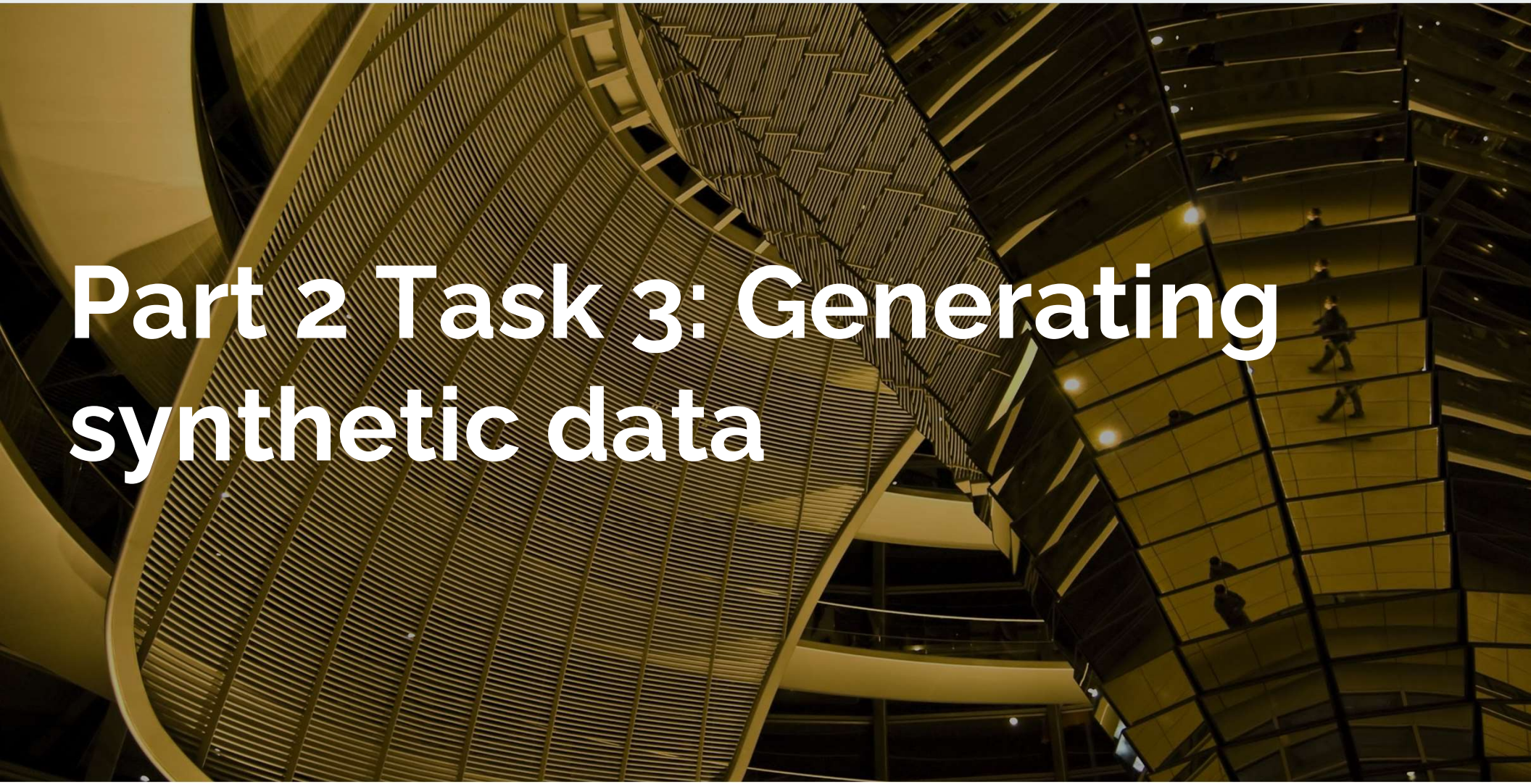
u_feedLifeCycle	User engagement on news feeds
u_newsCatInterestsST	User's short-term interest categories
e_m	Event source device model

Can I Quantify the Correlation?

When increase X feature by 80%, how will it affect the click rate?



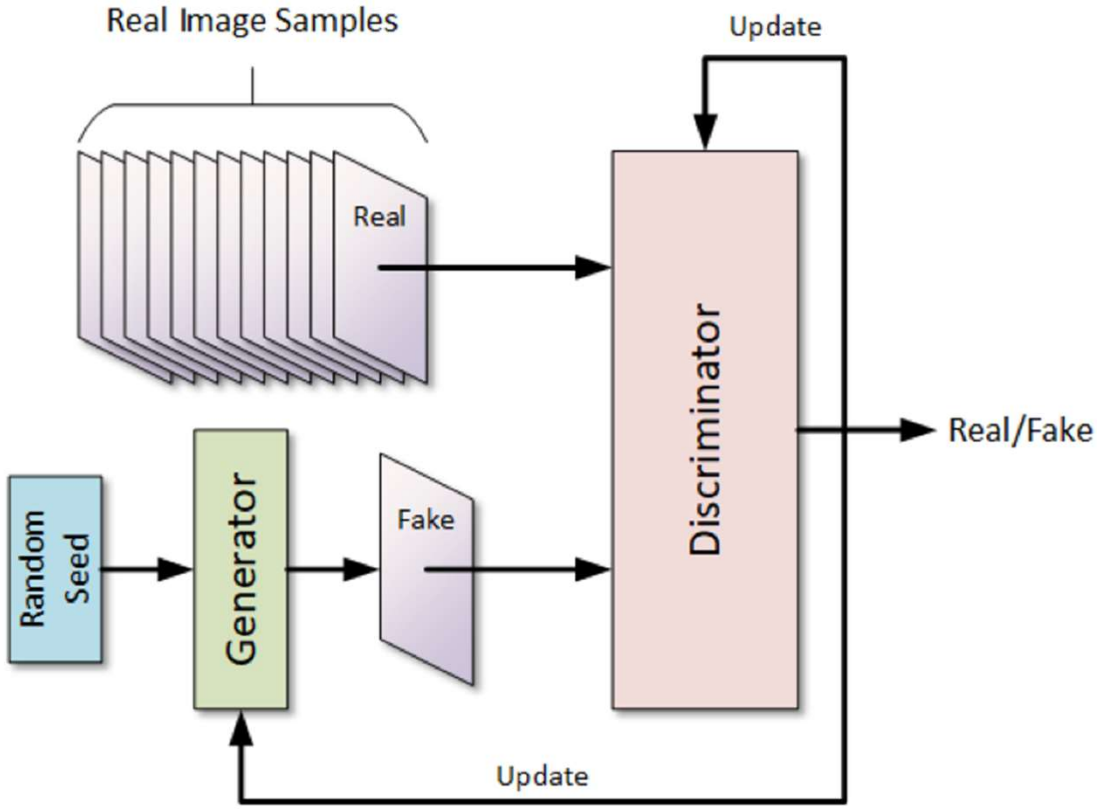




Part 2 Task 3: Generating synthetic data

Overview

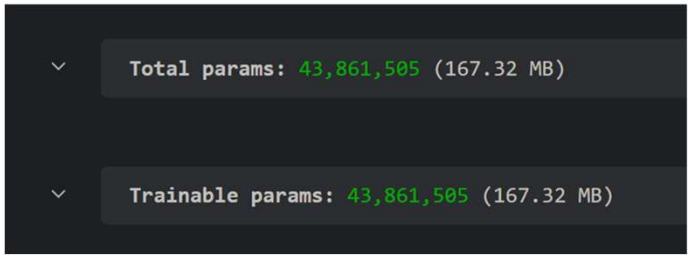
To create our synthetic data for confidentiality purposes a GAN model was used. This model took an input of the provided merged datasets. Only one dataset was made for the union of the two datasets as null points in non-clicked rows biased the model if set to 0. Due to the large size of the data sampling was done as to allow the model to train and not create memory errors. If possible the model can be adjusted to include the full data and increase number of epochs.



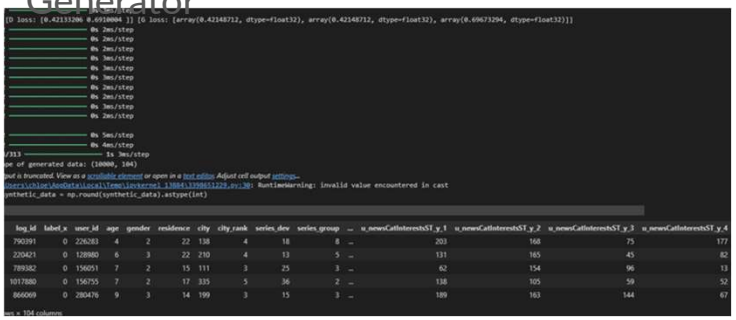
Generations

Several generations were made to try and perfect the model, this included models that had changes to items such as: learning rate, layers, nodes, type of GAN, and the input data. Some of the highlights of the models are here.

M1 treated all columns as categorical and used OneHotEncoder for each column creating an extremely high number of parameters.



M2 was changed to numerical input and had a high loss and low accuracy for both the Discriminator and the Generator



Generations cont.

M4 increased to 3 layers for the Generator and 2 for the discriminator. This model was plagued with an issue of destabilizing around 100-200 epochs and began to diverge, decreasing accuracy and increasing loss.

M7 was a form of a Wasserstein GAN in attempts to combat mode collapse that was present in some of the synthetically generated data. It's discriminator and generator diverged and the idea had to be scrapped.

Final Model

The final model that was resulted was a 2 layer GAN whos strength lied in the efficiency. It did not diverge and had small enough inputs to run over many epochs. After 1000 epochs it had a resulting loss of 3% on each model and 99.8% accuracy.

Layer (type)	Output Shape	Param #
dense_84 (Dense)	(None, 256)	389,888
leaky_re_lu_42 (LeakyReLU)	(None, 256)	0
dense_85 (Dense)	(None, 1)	257

Summary of Discriminator ^
and Generator (below)

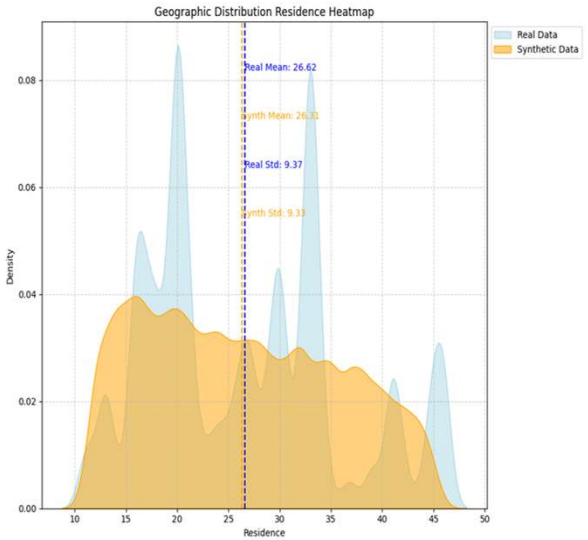
Layer (type)	Output Shape	Param #
dense_86 (Dense)	(None, 256)	25,856
leaky_re_lu_43 (LeakyReLU)	(None, 256)	0
batch_normalization_21 (BatchNormalization)	(None, 256)	1,024
dense_87 (Dense)	(None, 1522)	391,154

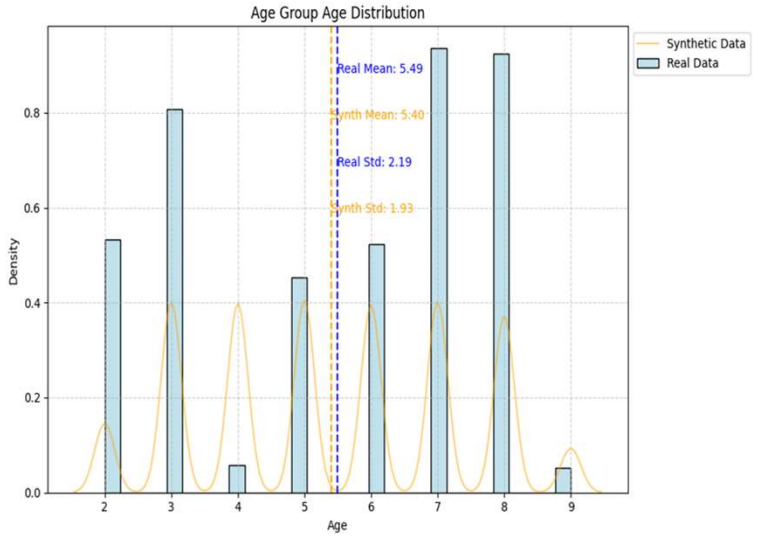
```
997 [D loss: [0.03034906 0.99878633]] [G loss: [array(0.03034652, dtype=float32), array(0.03034652,
dtype=float32), array(0.9987866, dtype=float32)]]
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
998 [D loss: [0.03033113 0.9987875 ]] [G loss: [array(0.03032867, dtype=float32), array(0.03032867,
dtype=float32), array(0.9987879, dtype=float32)]]
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
999 [D loss: [0.03031115 0.9987888 ]] [G loss: [array(0.03030852, dtype=float32), array(0.03030852,
dtype=float32), array(0.9987891, dtype=float32)]]
3125/3125 ━━━━━━━━━━━━━━━━━━━━ 3s 1ms/step
Shape of generated data: (100000, 104)
```

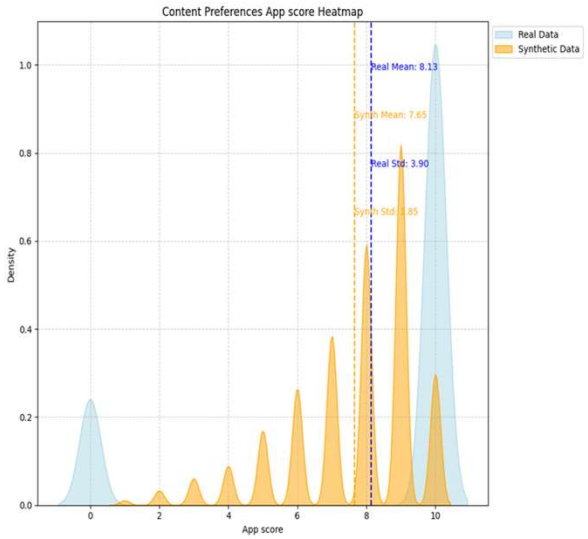
Future improvements:

- M4 had worse accuracy and loss but better data gen
- Also generate the non click data

Real vs Synthetic







- The synthetic data closely mirrors the real data across age groups, geographic distribution, and content preferences.

- **Age Groups:** Synthetic data mean = 5.40, std dev = 1.93 vs. real data mean = 5.49, std dev = 2.19.
- **Geographic Distribution (Residence):** Synthetic data mean = 26.31, std dev = 9.33 vs. real data mean = 26.62, std dev = 9.37.
- **Content Preferences (App Score):** Synthetic data mean = 7.65, std dev = 1.85 vs. real data mean = 8.13, std dev = 3.90.

Thank you.