



# Advertisement CTR Prediction

## Exploratory Data Analysis

DS5220 / Fall 2023 Semester

Team Members: Liyang Song, Qian Yin

Oct 29, 2023

# Updates on the source dataset

---

- As the original dataset contains over **40 million** observations, it could be expensive to run a **GridSearch** or complex modeling (e.g. **Random Forest**) in the afterward modeling phase.
- After communicating with the professor, we would only randomly resample **2.5%** of all observations (around **1 million**) from the original set for our project.
- Also, we have uploaded the resampled dataset on our **AWS S3**, which can be downloaded through *download\_data.ipynb*, and split into train and test set through *train\_test\_split.ipynb*.
- We use GitHub for version control and collaborative development. Our repository: <https://github.com/LiyangSong/Advertisement-CTR-Prediction>

# Reproducibility

---

- Submitted files include Python module files (**.py** ), Jupyter Notebooks (**.ipynb** and **.html**), an environment file (**.yml**), and PowerPoints (**.pdf**).
- Run *download\_data.ipynb* to download resampled dataset.
- Run *train\_test\_split.ipynb* to implement train and test set split.
- Run *eda.ipynb* to implement exploratory data analysis (EDA).
- Run *prep.ipynb* to try data transformation and EDA on the transformed dataset.

# EDA on Train Set: General information

```
df.head():
```

	uid	task_id	adv_id	creat_type_cd	adv_prim_id	dev_id	inter_type_cd	slot_id	spread_app_id	tags	...
0	1641431	5177	1998	7	191	60	5	21	82	14	...
1	2021896	4628	4530	7	177	56	5	17	31	40	...
2	1790795	2709	1413	7	134	55	4	17	65	18	...
3	1216709	1949	6143	7	150	17	5	21	11	39	...
4	1635521	4806	2176	7	206	64	5	15	22	39	...

```
df.describe:
```

	uid	task_id	adv_id	creat_type_cd	adv_prim_id	dev_id	inter_type_cd
count	8.381420e+05	838142.000000	838142.000000	838142.000000	838142.000000	838142.000000	838142.000000
mean	1.618734e+06	3436.778738	3964.562067	6.491097	159.346748	41.596531	4.647426
std	3.574750e+05	1430.118430	1720.086656	1.230525	30.895882	17.418631	0.709892
min	1.000004e+06	1001.000000	1001.000000	2.000000	101.000000	11.000000	2.000000
25%	1.309340e+06	2229.000000	2504.000000	6.000000	134.000000	29.000000	5.000000
50%	1.618150e+06	3370.000000	4056.000000	7.000000	156.000000	37.000000	5.000000
75%	1.929031e+06	4595.000000	5461.000000	7.000000	180.000000	60.000000	5.000000
max	2.237671e+06	5992.000000	7020.000000	9.000000	214.000000	72.000000	5.000000

```
24 emui_dev      838142 non-null int64
25 list_time     838142 non-null int64
26 device_price  838142 non-null int64
27 up_life_duration 838142 non-null int64
28 up_membership_grade 838142 non-null int64
29 membership_life_duration 838142 non-null int64
30 consume_purchase 838142 non-null int64
31 communication_onlinerate 838142 non-null object
32 communication_avgonline_30d 838142 non-null int64
33 indu_name     838142 non-null int64
34 pt_d         838142 non-null int64
35 label        838142 non-null int64
dtypes: int64(35), object(1)
```

```
df.shape:
(838142, 36)
```

# EDA on Train Set: Duplication and missingness

```
eda_utils.check_out_duplicate_obs(train_df_eda)
```

Check out duplicate observations:

df.shape: (838142, 36)

drop\_dup\_df.shape: (833795, 36)

Caution: data set contains duplicate observations!!!

```
: eda_utils.check_out_missing_target(train_df_eda, target)
```

Check out observations with missing target:

df.shape: (838142, 36)

drop\_miss\_tar\_df.shape: (838142, 36)

No missing-target observations observed in data set.

```
eda_utils.check_out_missingness(train_df_eda)
```

Check out missingness:

No missing values in data set.

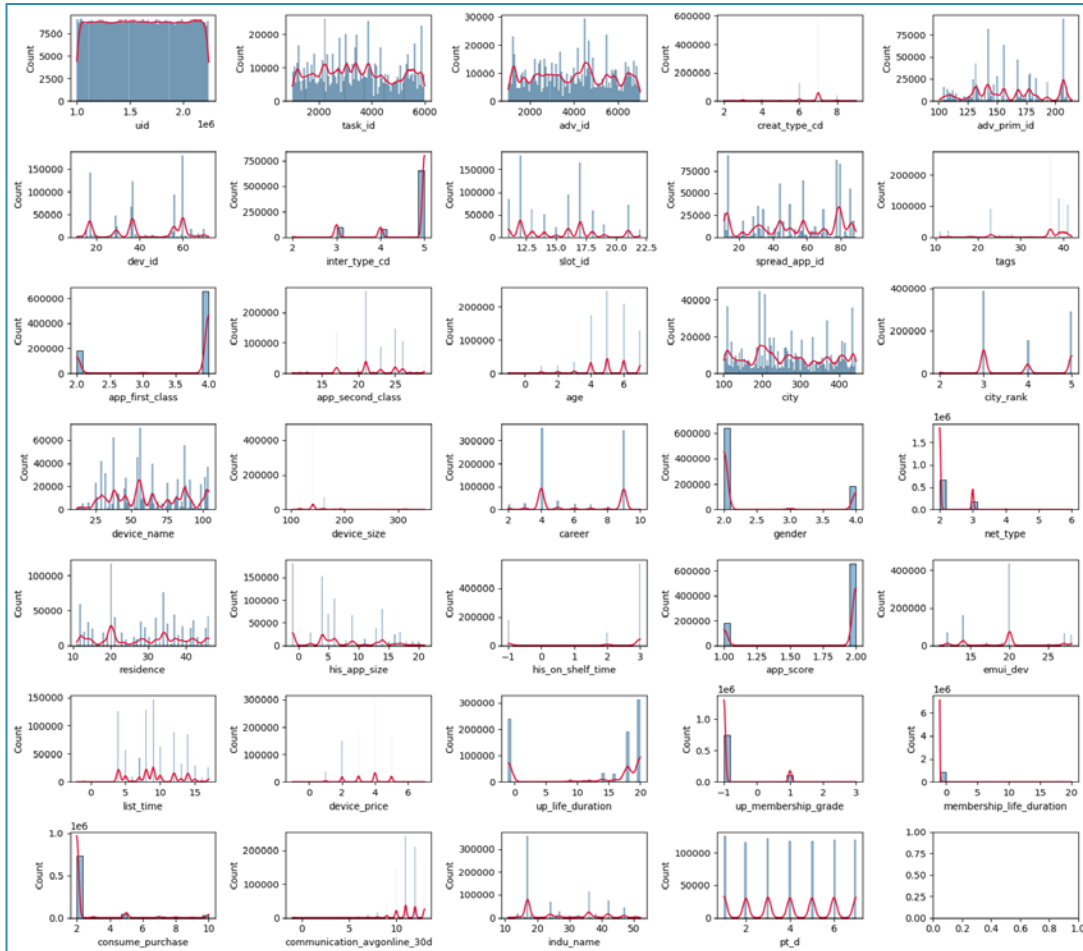
- There are **duplicate observations** in the set, which should be dropped at the beginning of the data transformation.
- **No missing value** in both attributes and the target variable.

# EDA on Train Set: Identify numerical and categorical attributes

---

- Based on the data type of attribute columns, there is only one categorical attribute *communication\_onlinerate*. However, some 'numerical' attributes seem to be categorical from the field explanation (see *data\_fields.json* in repo) and domain knowledge.
- After checking the original dataset, it seems the data has been **label-encoded**, which means categories in categorical attributes are replaced with numbers.
- These pre-encoded categorical attributes have no statistical meaning with their number. These numbers can be actually seen as another type of 'label'.
- In addition, directly label-encoding categorical attributes may be not suitable sometimes. It could be suitable if the attribute is ordinal, yet label-encoding nominal attributes is meaningless, as there is no actual numerical relationship between them.
- Thus, we will look at the distribution of all 'numerical' attributes and **find actually categorical ones** within them.

# EDA on Train Set: Identify numerical and categorical attributes



- The histogram plots reveal some **possible pre-encoded attributes**.
- Some are clearly nominal: IDs(*uid*, *task\_id*, etc), city, career, gender, etc.
- Some are like ordinal: *age*, *city\_rank*, etc. However, they may also be pre-aggregated. For instance, there are only 8 unique values of *age*, which is not realistic.
- After communicating with the professor, we would treat all categorical attributes as the same type rather than splitting into nominal and ordinal ones.
- After looking into the original dataset, **all attributes are pre-encoded and no one can be defined as numerical**. It may be for the consideration of privacy.

# EDA on Train Set: Target encode categorical attributes

- As the high number of unique values in some attributes, **Target-Encoding** rather than OneHot-Encoding should be applied.

```
train_df_eda_encoded = data_prepare_utils.target_encode_categorical(train_df_eda, categorical_attr_list, target)
```

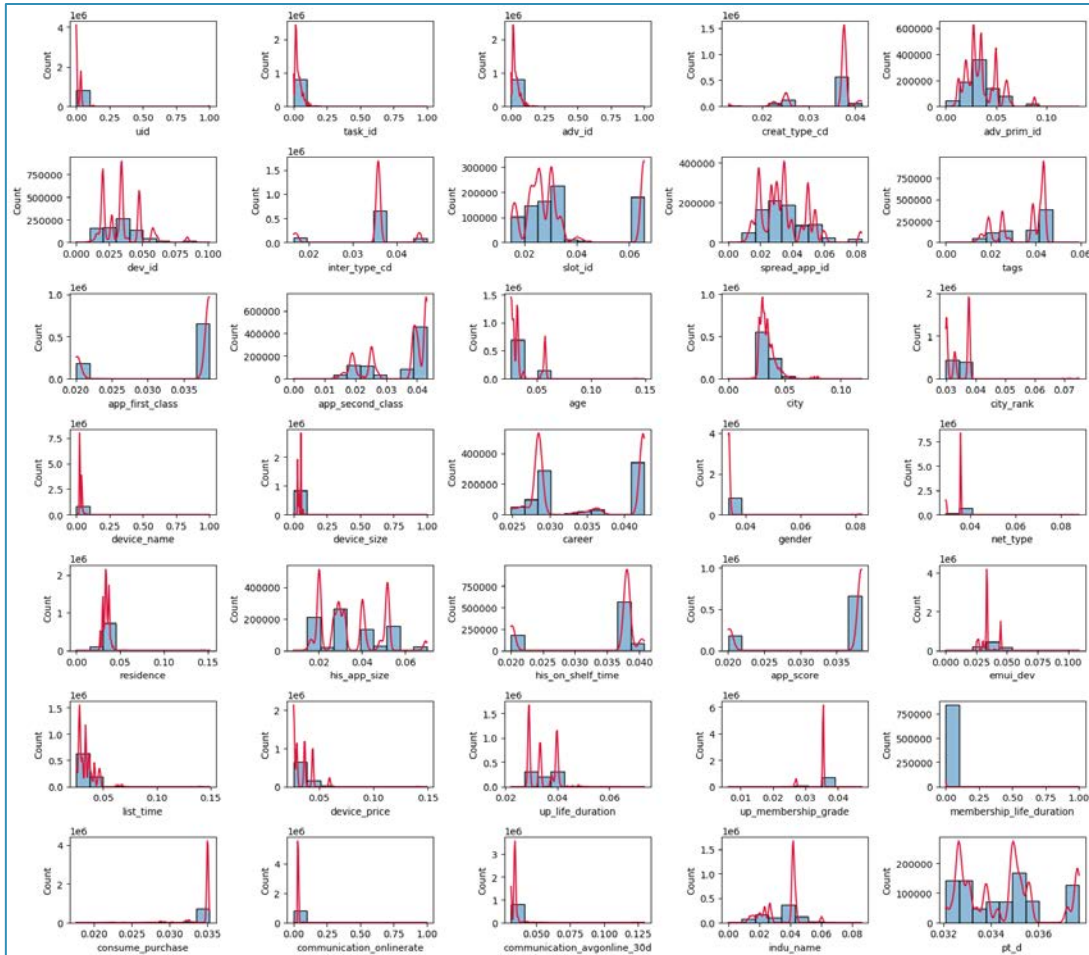
Target encode categorical attributes:

encoded\_df.head():

	uid	task_id	adv_id	creat_type_cd	adv_prim_id	dev_id	inter_type_cd	slot_id	spread_app_id	tags	...
0	0.000000	0.038679	0.038679	0.037668	0.029137	0.020065	0.035876	0.025592	0.029137	0.029137	...
1	0.034493	0.012218	0.012218	0.037681	0.032378	0.034536	0.035878	0.030219	0.030939	0.038545	...
2	0.034493	0.033665	0.033665	0.037665	0.033763	0.033763	0.045227	0.030463	0.033763	0.028481	...
3	0.000000	0.008956	0.008956	0.037665	0.035366	0.047814	0.035886	0.025305	0.035366	0.042652	...
4	0.000000	0.053058	0.053058	0.037665	0.038609	0.039587	0.035886	0.016242	0.038609	0.042652	...

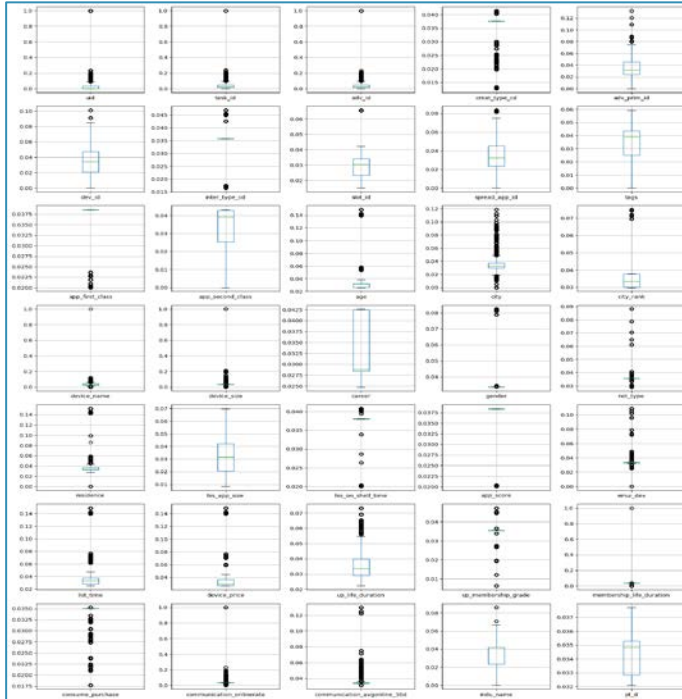


# EDA on Train Set: EDA on encoded attributes (distribution)



- The distribution of encoded attributes becomes **consecutive and smoother** than before, as target encoding is less sensitive to extreme values.

# EDA on Train Set: EDA on encoded attributes (outlier detection)

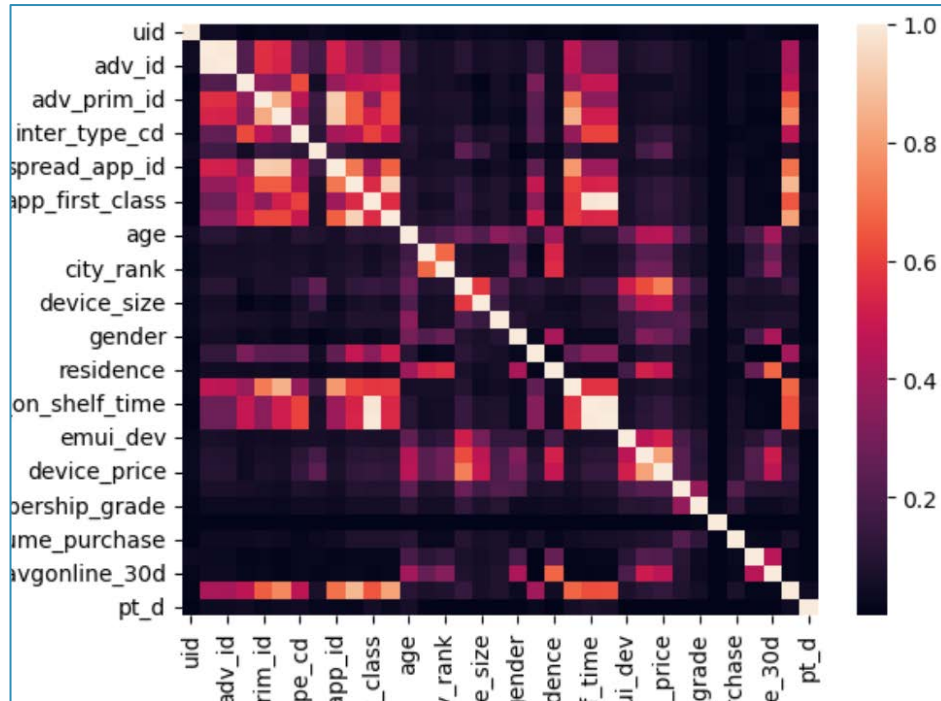


Implement Tukey's fences to identify outliers based on the Inter Quartile Range (IQR) method:

	Attribute	Outliers Prob Count	Outliers Prob Fraction	Outliers Poss Count	Outliers Poss Fraction
16	device_size	364305	0.434658	364845	0.435302
24	emui_dev	296287	0.353505	360704	0.430361
22	his_on_shelf_time	267541	0.319207	267541	0.319207
3	creat_type_cd	261764	0.312315	261764	0.312315
6	inter_type_cd	184833	0.220527	184833	0.220527
10	app_first_class	184675	0.220339	184675	0.220339
23	app_score	180435	0.215280	180435	0.215280
19	net_type	171692	0.204848	171692	0.204848
18	gender	158172	0.188717	195166	0.232856
12	age	151191	0.180388	151191	0.180388
30	consume_purchase	104001	0.124085	104001	0.124085
28	up_membership_grade	101984	0.121679	101984	0.121679
32	communication_avgonline_30d	39365	0.046967	225254	0.268754
31	communication_onlinerate	24121	0.028779	70296	0.083871

- There are still some attributes with a significantly high outlier fraction ( $>20\%$ ) after target encoding, this is reasonable for transformed categorical variables.
- We will not decide to drop them simply in this step. Considering the domain knowledge, some of them may be useful predictors.

# EDA on Train Set: EDA on encoded attributes (feature correlation)



Matrix visualizing correlation (>0.7) between n

	correlation
his_on_shelf_time with app_score	0.994035
task_id with adv_id	0.990200
app_first_class with app_score	0.988605
app_first_class with his_on_shelf_time	0.982526
tags with app_second_class	0.929792
adv_prim_id with spread_app_id	0.921037
dev_id with spread_app_id	0.907717
tags with indu_name	0.863911
dev_id with his_app_size	0.846389
adv_prim_id with dev_id	0.833137
list_time with device_price	0.815580
app_second_class with indu_name	0.814459
spread_app_id with his_app_size	0.791584
dev_id with indu_name	0.753021

	attribute	vif
24	app_score	133.794539
23	his_on_shelf_time	87.679437
2	task_id	51.808286
3	adv_id	51.275208
11	app_first_class	47.327007
9	spread_app_id	15.577611
10	tags	11.949464
6	dev_id	11.120764
12	app_second_class	8.287784
5	adv_prim_id	7.862803
34	indu_name	6.486875
27	device_price	5.075470
22	his_app_size	4.207545

- There are a remarkable number of attribute pairs with a **high correlation** (>0.7). We will consider dropping one from the pair of them.
- There are some very **high VIFs** (>5), which reveals the existence of multi co-linearity.

# EDA on Train Set: EDA on encoded attributes (feature usefulness)

Calculate the correlation between each attribute and the target:

	correlation
membership_life_duration	0.000068
pt_d	0.008201
consume_purchase	0.008991
net_type	0.014096
up_membership_grade	0.014871
communication_onlinerate	0.018961
city	0.025930
city_rank	0.026971
up_life_duration	0.028191
gender	0.028615
communication_avgonline_30d	0.031160
emui_dev	0.033253

Calculate the variance of each attribute:

	variance
membership_life_duration	0.000002
pt_d	0.000003
consume_purchase	0.000003
net_type	0.000007
up_membership_grade	0.000008
city_rank	0.000025
gender	0.000027
up_life_duration	0.000029
communication_avgonline_30d	0.000034
emui_dev	0.000038
creat_type_cd	0.000041

- The correlation of each attribute with the target is **not high** enough. This is reasonable for a large-number attributes model.
- The variance for all attributes is **quite tiny** with the biggest one  $< 0.01$ . This may due to the highly unbalanced target variable, or the high cardinality of attributes.
- Low-variance attributes do not represent useless as they may still carry useful information.

# Identify the promising transformations

- Deal with duplicate observations.
- No missingness in data set.
- Target encode categorical attributes.
- Drop highly correlated attributes.
  - *app\_score*. A very high VIF, very high correlations, and a high outlier fraction.
  - *his\_on\_shelf\_time*. A very high VIF, very high correlations, and a high outlier fraction.
  - *task\_id*. A very high VIF and a very high correlation.
  - *spread\_app\_id*. A high VIF and very high correlations.
  - *tags*. A high VIF and very high correlations.
  - *dev\_id*. A high VIF and very high correlations.
  - *app\_second\_class*. A high VIF and very high correlations.
  - *adv\_prim\_id*. A high VIF and very high correlations.
  - *device\_price*. A high VIF and very high correlations.

```
attrs_to_drop  
  
['app_score',  
 'his_on_shelf_time',  
 'task_id',  
 'spread_app_id',  
 'tags',  
 'dev_id',  
 'app_second_class',  
 'adv_prim_id',  
 'device_price']
```