CLICK THROUGH RATE (CTR) PREDICTION

- Vanraj Dinesh Bohra
- Email : vanrajbohra@gmail.com

- August 2022

AGENDA

- Business Problem Statement
- EDA
- Model Building
- Feature Engineering
- Model improvement
    - Gradient Boosting Classifier
    - AdaBoost Classifier
    - Logistic Regression
- Model Evaluation

Business Problem Statement

Problem Statement:

1. CTR is an important exercise companies need to do before making any decision on Online Advertising.
2. CTR is a very important metric for evaluating ad performance.
3. The objective is to predict whether the audience will click on an ad or not and thus help the marketing team answer ad placement-related questions.

Analysis Objective:

1. Perform EDA on the dataset provided.
2. Implement and evaluate different Classification model and assess model performance.
3. Use model simplification techniques like feature creation and selection, to improve model predictability.

Data to Analyze:

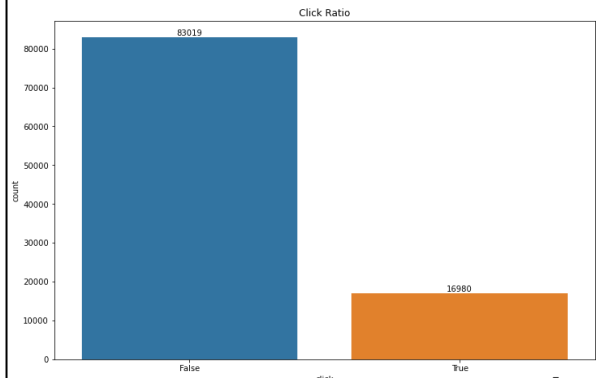
1. 99999 record with classifier where user have clicked on add or not.
2. 25 other attribute such as site detail, app detail, device detail and 8 other anonymous attribute is available in data to analyse

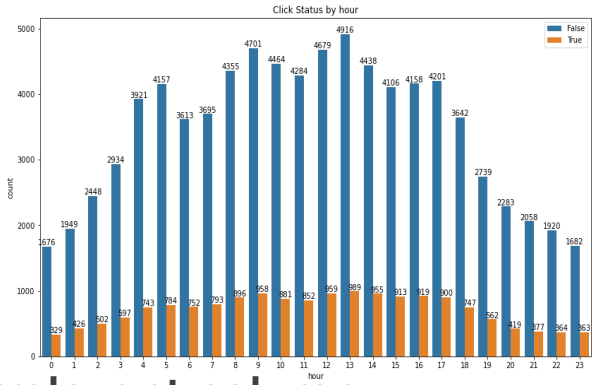
EDA

1. Data Cleansing:

- Checked Duplicate column if available in given dataset
- Null handling

2. Univariate and Bivariate Analysis*:





3. Dropping redundant / irrelevant column:

- Dropped month, device_id,device_ip site_id, app_id in order to keep data simple and we were having another supplementary feature available in dataset for this

4. Train Test Split :

- Data is split into 70-30 split as a train and test dataset.

	0	1	2	3	4
click	False	True	False	False	True
C1	1005	1005	1005	1002	1005
banner_pos	1	1	0	0	0
site_id	856e6d3f	e151e245	e3c09f3a	0da94452	1fbe01fe
site_domain	58a89a43	7e091613	d262cf1e	248e439f	f3845767
site_category	f028772b	f028772b	28905ebd	50e219e0	28905ebd
app_id	ecad2386	ecad2386	ecad2386	ecad2386	ecad2386
app_domain	7801e8d9	7801e8d9	7801e8d9	7801e8d9	7801e8d9
app_category	07d7df22	07d7df22	07d7df22	07d7df22	07d7df22
device_id	a99f214a	a99f214a	a99f214a	0fa578fd	a99f214a
device_ip	962c8333	5b1f94b9	a9a84f4c	88c62dad	1e5e0d0e
device_model	be6db1d7	1b13b020	9a45a8e8	ea6abc60	36d749e5
device_type	1	1	1	0	1
device_conn_type	0	0	0	0	0
C14	22683	17037	22155	21591	15708
C15	320	320	320	320	320
C16	50	50	50	50	50
C17	2528	1934	2552	2478	1722
C18	0	2	3	3	0
C19	39	39	167	167	35
C20	100075	-1	100202	100074	-1
C21	221	16	23	23	79
month	10	10	10	10	10
dayofweek	1	2	3	2	1
day	28	22	23	22	21
hour	14	19	18	19	8
y	0	1	0	0	1

	Unique Value Count
click	2
C1	7
banner_pos	7
site_id	1485
site_domain	1331
site_category	19
app_id	1354
app_domain	96
app_category	21
device_id	16801
device_ip	78013
device_model	3145
device_type	4
device_conn_type	4
C14	1722
C15	8
C16	9
C17	399
C18	4
C19	64
C20	154
C21	60
month	1
dayofweek	7
day	10
hour	24
y	2

**Univariate and Bivariate Analysis is done for all the variables, Please refer python notebook for more detail*

Model Building

1. **One Hot Encoding / Dummy Encoding :**
   - Dummy encoding on dataset resulted in 22 column converted into 7075 column
   - A dummy variable is a binary variable that indicates whether a separate categorical variable takes on a specific value.
   - For example, if a banner_pos have 7 unique value, then 7 different column will be created with value 1 and 0.
2. **Train-Test Data Split:** Data is split into Train and Test in 70-30 ratio.
3. **Scaling:** Standardized the train and test data by scaling.
4. **Model Build:** Model is built, and result is for following algorithm:

model	run_time	tp	tn	fp	fn	correct	incorrect	accuracy	precision	recall	f1	roc_auc	avg_precision
DummyClassifier_most_frequent	0.43	0	24906	0	5094	24906	5094	0.8302	0.0	0.0	0.0	0.5	0.17
LogisticRegression	7.5	529	24160	746	4565	24689	5311	0.822967	0.414902	0.103848	0.166	0.647	0.195
RandomForestClassifier	23.48	591	24070	836	4503	24661	5339	0.822033	0.414156	0.116019	0.181	0.683	0.198
DecisionTreeClassifier	9.68	1176	22070	2836	3918	23246	6754	0.774867	0.293121	0.23086	0.258	0.55	0.198

5. **Result :**
   - Accuracy and ROC_AUC doesn’t seem to be great for classification model
   - Dummy encoding of feature columns leads to high cardinality.
   - It doesn't make sense to model with such complex number of columns due to poor interpretability and high computational time.¶
   - So, its better to encode columns based on statistics as it will result in fewer columns and will be interpretable.

Feature Engineeeing:

1. **Feature Engineering done by creating ratio of Clicked vs Non-Clicked and Quantiles:**
    1. Impression are total click per category
    2. CTR is the ratio of Clicked over total impression.
    3. Non Click ratio is Non-Clicked over total impression.
    4. we find odd ratio such as CTR / Non-Click Ration
    5. Quantile category 3 has above 75% of the CTR values
    6. Quantile category 2 has above 50% of the CTR values
    7. Quantile category 1 has above 25% of the CTR values
    8. Quantile category 0 has less than 25% of the CTR values
2. **We ran Machine Learning model on this prepared data**
3. **Result shows Gradient Boosting Classifier, AdaBoost Classifier and Logistic Regression clearly outperformed:**

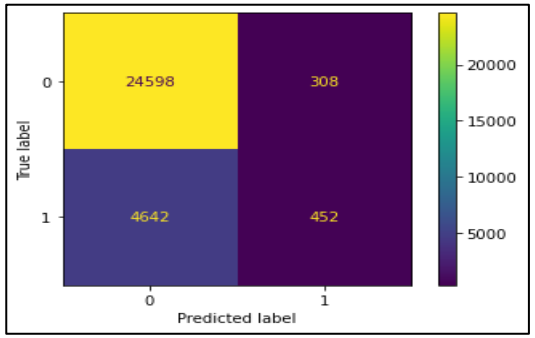
model	run_time	tp	tn	fp	fn	correct	incorrect	accuracy	precision	recall	f1	roc_auc	avg_precision
GradientBoostingClassifier	0.97	680	24568	338	4414	25248	4752	0.8416	0.667976	0.13349	0.223	0.772	0.236
AdaBoostClassifier	0.29	622	24570	336	4472	25192	4808	0.839733	0.649269	0.122104	0.206	0.765	0.228
LogisticRegression	0.17	415	24630	276	4679	25045	4955	0.834833	0.600579	0.081468	0.143	0.74	0.205
DummyClassifier_most_frequent	0.0	0	24906	0	5094	24906	5094	0.8302	0.0	0.0	0.0	0.5	0.17
RandomForestClassifier	0.71	1005	23533	1373	4089	24538	5462	0.817933	0.422624	0.197291	0.269	0.715	0.22
BaggingClassifier	0.33	978	23382	1524	4116	24360	5640	0.812	0.390887	0.191991	0.258	0.691	0.212
DecisionTreeClassifier	0.05	1557	21208	3698	3537	22765	7235	0.758833	0.296289	0.305654	0.301	0.577	0.208

Model Improvement : Logistic Regression(Hyperparameter Tuning)

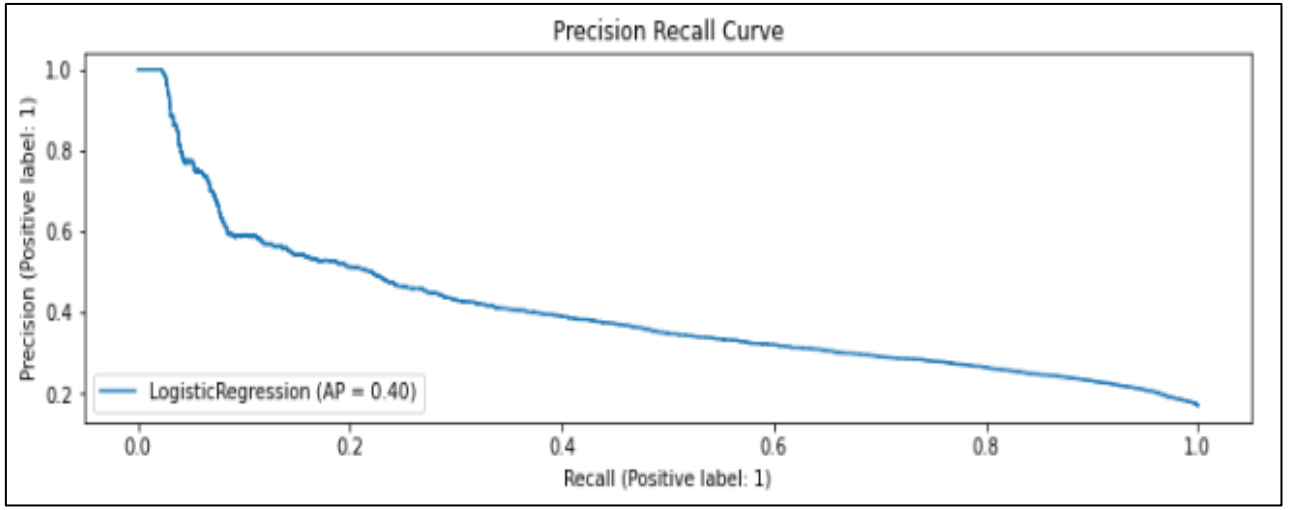
Logistic Regression:

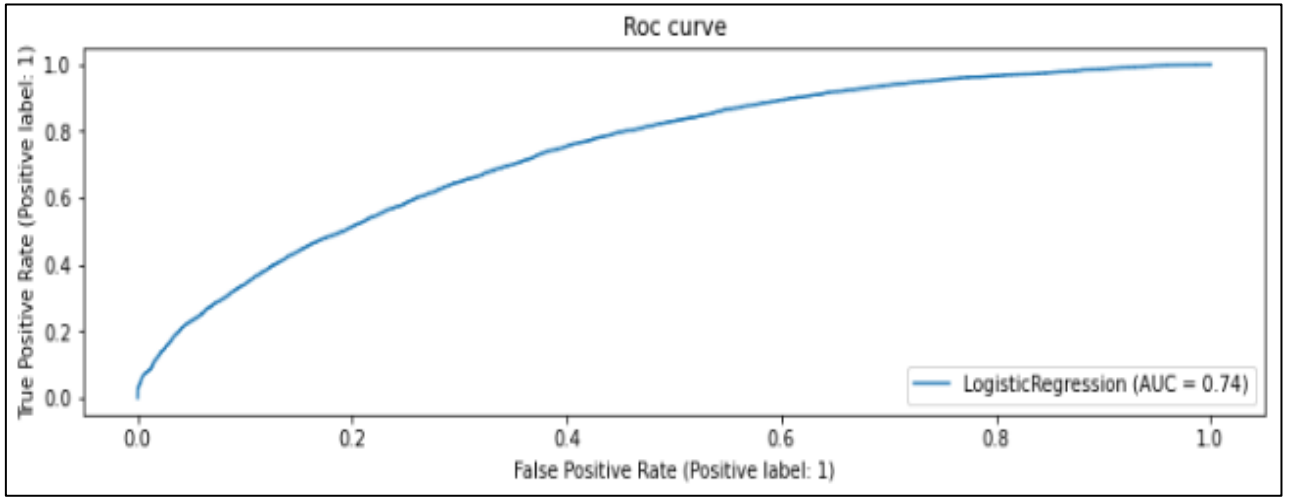
- It is valuable for predicting the likelihood of an event
- Since there are two classes of the dependent variable(click feature), logistic regression help in determining their probabilities.
- It's mainly used when the prediction is categorical, for example, yes or no, true or false, 0 or 1.
- **Hyperparameter Tunning :**
  - Inverse of regularization strength(C) set to 100
  - Penalty is set to 12
  - Above value we got from Grid Search SV (cross-validated grid-search)

	precision	recall	f1-score	support
0	0.84	0.99	0.91	24906
1	0.59	0.09	0.15	5094
accuracy			0.83	30000
macro avg	0.72	0.54	0.53	30000
weighted avg	0.80	0.83	0.78	30000



model	run_time	tp	tn	fp	fn	correct	incorrect	accuracy	precision	recall	f1	roc_auc	avg_precision
LogisticRegression	0.17	415	24630	276	4679	25045	4955	0.834833	0.600579	0.081468	0.143	0.74	0.205
Logistic Regression with Hypertunning	0.05	452	24598	308	4642	25050	4950	0.835	0.594737	0.088732	0.154	0.741	0.208



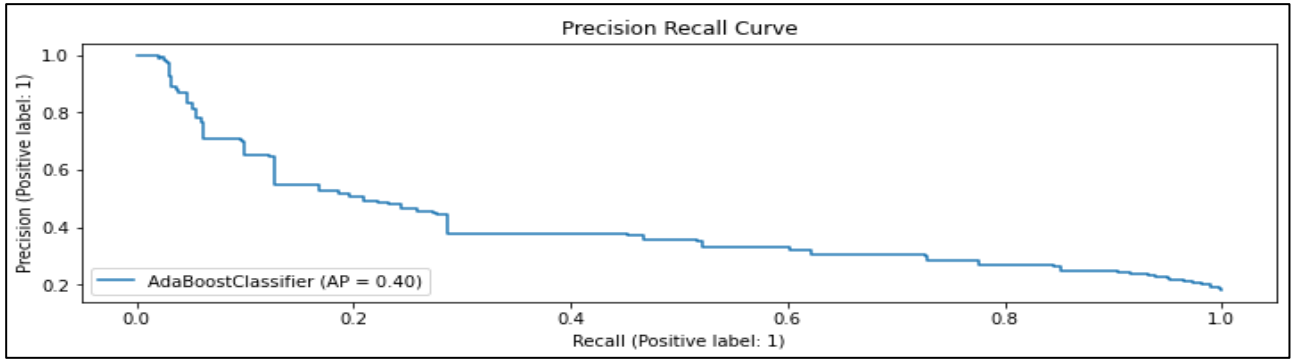


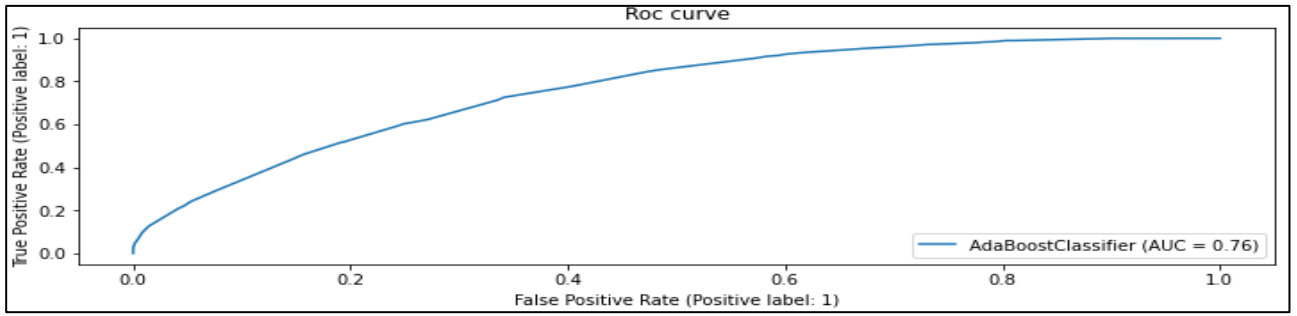
Model Improvement : AdaBoost Classifier(Hyperparameter Tuning)

AdaBoost Classifier:

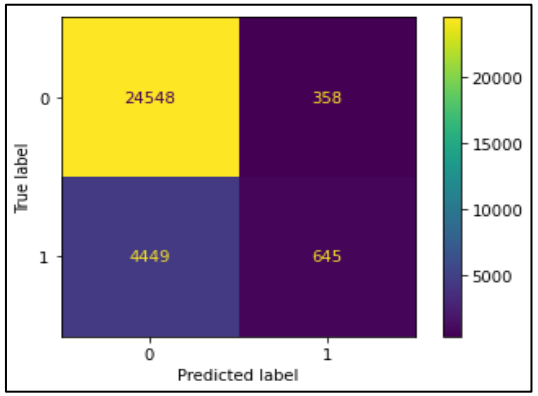
- AdaBoost is best used to boost the performance of decision trees on binary classification problems.
- **Hyperparameter Tunning:**
  - n_estimator = 28
    - The maximum number of estimators at which boosting is terminated.
  - learning_rate = 1
    - Above value we got from Grid Search SV (cross-validated grid-search)
- Accuracy percentage slightly only increase after hyper parameter Tunning
- Based on Feature Coefficient, we can select important feature as device model(odd), site domain(odd), C14(odd), app domain(odd) and app category(odd)

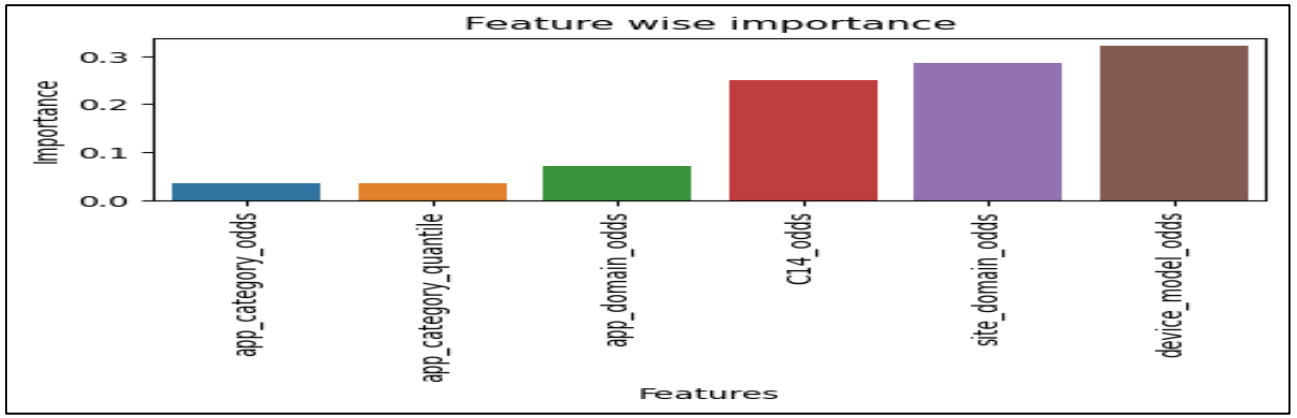
model	run_time	tp	tn	fp	fn	correct	incorrect	accuracy	precision	recall	f1	roc_auc	avg_precision
AdaBoost Classifier	0.3	622	24570	336	4472	25192	4808	0.839733	0.649269	0.122104	0.206	0.765	0.228
AdaBoost Classifier with tuned hyperparameters	0.17	645	24548	358	4449	25193	4807	0.839767	0.643071	0.12662	0.212	0.762	0.23





	precision	recall	f1-score	support
0	0.85	0.99	0.91	24906
1	0.64	0.13	0.21	5094
accuracy			0.84	30000
macro avg	0.74	0.56	0.56	30000
weighted avg	0.81	0.84	0.79	30000



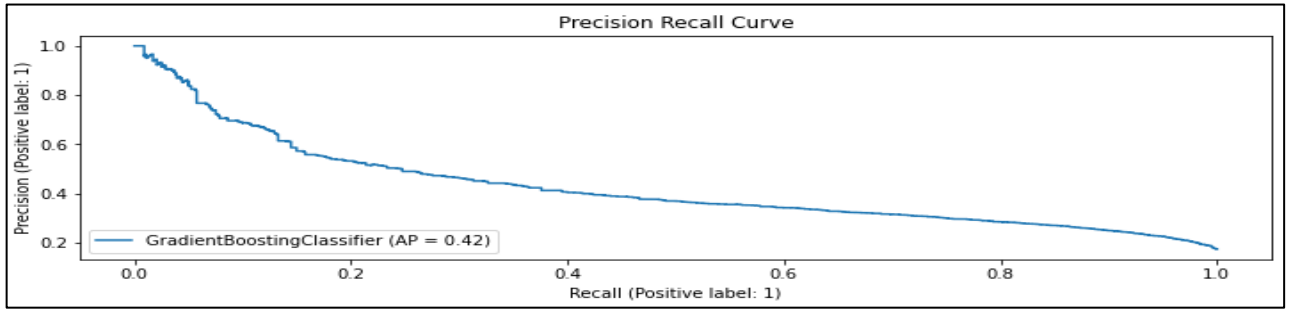


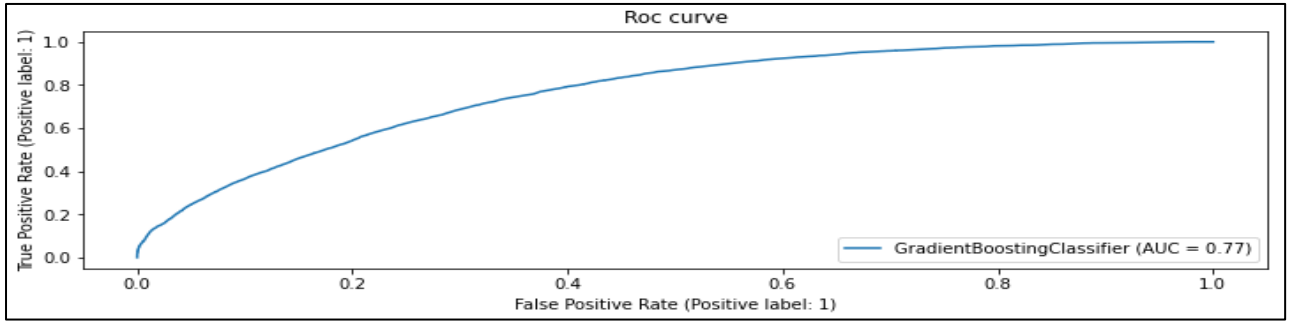
Model Improvement : Gradient Boost Classifier(Hyperparameter Tuning)

Gradient Boost Classifier:

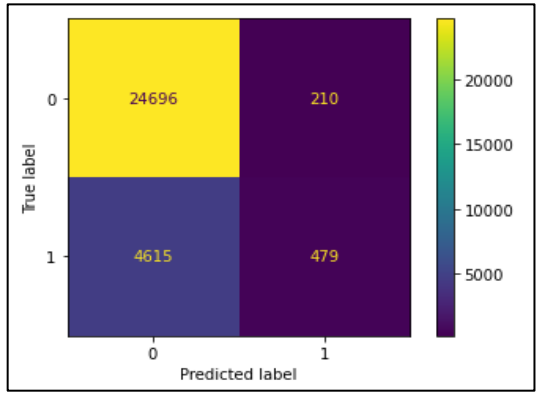
- AdaBoost is best used to boost the performance of decision trees on binary classification problems.
- Iteratively learns from each of the weak learners to build a strong model.
- Lots of flexibility - can optimize on different loss functions and provides several hyper parameter tuning options that make the function fit very flexible.
- **Hyperparameter Tunning :**
  - n_estimator = 28
    - The maximum number of estimators at which boosting is terminated.
  - learning_rate = 1
    - Above value we got from Grid Search SV (cross-validated grid-search)
  - Above value we got from Grid Search SV (cross-validated grid-search)
- Based on feature coefficient, we can select the important features with positive coefficient - c14, site_domain, device_model

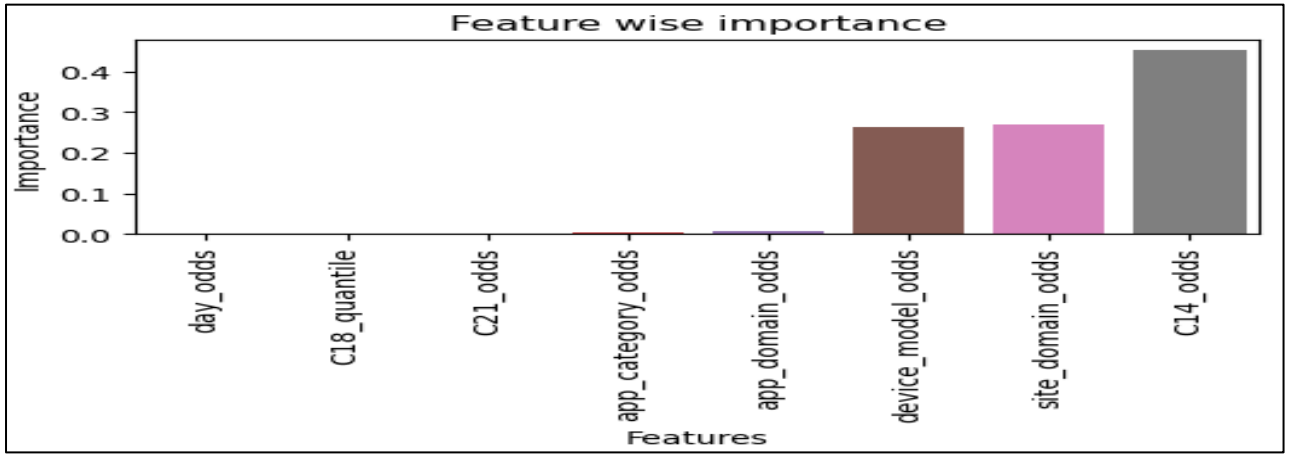
	model	run_time	tp	tn	fp	fn	correct	incorrect	accuracy	precision	recall	f1	roc_auc	avg_precision
	Gradient Boosting Classifier	0.93	680	24568	338	4414	25248	4752	0.8416	0.667976	0.13349	0.223	0.772	0.236
	Gradient boosting classifier with tuned hyperp...	0.28	479	24696	210	4615	25175	4825	0.839167	0.69521	0.094032	0.166	0.766	0.219





	precision	recall	f1-score	support
0	0.84	0.99	0.91	24906
1	0.70	0.09	0.17	5094
accuracy			0.84	30000
macro avg	0.77	0.54	0.54	30000
weighted avg	0.82	0.84	0.78	30000





Model Evaluation:

we are selecting Gradient Boost Classifier algorithm for following reason:

1. **Gives highest ROC-Accuracy score(Area under curve of ROC curve works as a good measure of model performance).**
2. **Good Accuracy Score : 84.16%.**
3. **Best Correct prediction : 25248 correct**
4. **Average Precision score is also best**

model	run_time	tp	tn	fp	fn	correct	incorrect	accuracy	precision	recall	f1	roc_auc	avg_precision
GradientBoostingClassifier	0.97	680	24568	338	4414	25248	4752	0.8416	0.667976	0.13349	0.223	0.772	0.236
Gradient boosting classifier with tuned hyperp...	0.28	479	24696	210	4615	25175	4825	0.839167	0.69521	0.094032	0.166	0.766	0.219
AdaBoostClassifier	0.3	622	24570	336	4472	25192	4808	0.839733	0.649269	0.122104	0.206	0.765	0.228
AdaBoost Classifier with tuned hyperparameters	0.16	645	24548	358	4449	25193	4807	0.839767	0.643071	0.12662	0.212	0.762	0.23
Logistic Regression with Hypertunning	0.04	452	24598	308	4642	25050	4950	0.835	0.594737	0.088732	0.154	0.741	0.208
LogisticRegression	0.11	415	24630	276	4679	25045	4955	0.834833	0.600579	0.081468	0.143	0.74	0.205
RandomForestClassifier	0.75	1002	23582	1324	4092	24584	5416	0.819467	0.430782	0.196702	0.27	0.715	0.221
BaggingClassifier	0.36	978	23382	1524	4116	24360	5640	0.812	0.390887	0.191991	0.258	0.691	0.212
DecisionTreeClassifier	0.05	1545	21222	3684	3549	22767	7233	0.7589	0.295468	0.303298	0.299	0.574	0.208
DummyClassifier_most_frequent	0.0	0	24906	0	5094	24906	5094	0.8302	0.0	0.0	0.0	0.5	0.17