ML FOR ADVERTISING IN DIFFERENT ZONES TO TARGET CUSTOMERS

TEAM MEMBERS AND RESPONSIBILITIES

Aman Khandelwal	U101116FCS007	Training and Analysis of dataset
Chinju Mary George	U101116FCS025	Development Of Prediction model
Gaurav Mundhra	U101116FCS037	Data Preprocessing and evaluation of model
Shashwat M Shah	U101116FCS112	Webpage Development
Sherry Sharma	U101116FCS287	Analysis of Dataset
Abhijit Singh	U101116FCS002	Finding Dataset

RATIONALE OF WORK

- Advertising is a key factor for expansion of market in today’s world.This project helps in targeting customers of different zones with the help of Machine learning algorithms.

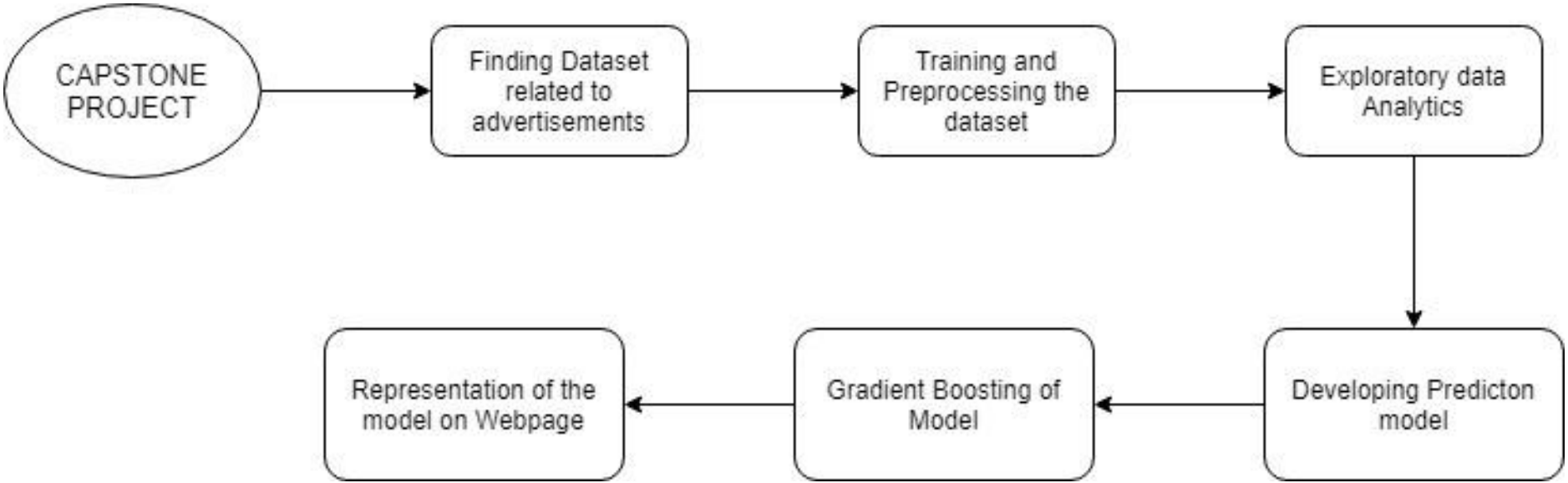
OBJECTIVE

- A model to make predictions for the probabilities of a person clicking on the ad(or not) using click through rate probability to analyze the dataset based on some significant features.

TOOLS USED

- Anaconda
- Docker
- PyBuilder
- GitHub
- Jenkins
- Visual Studio Code
- Google colab

WORKFLOW


CAPSTONE PROJECT
Finding Dataset related to advertisements
Training and Preprocessing the dataset
Exploratory data Analytics
Developing Predicton model
Gradient Boosting of Model
Representation of the model on Webpage

PROJECT PLAN REVISED

February

1. PPT submitted on project plan and roadmap
2. Collected dataset for different business organisation .
3. Studied machine learning algorithms useful for this project.

March

1. Implemented our algorithm based on the factors we want to analyse data.
2. Testing and editing algorithms with different data set according till the optimal outcome is retrieved.

April

1. Developed a model based on the analysis and the algorithm.
2. Developing a user friendly website with all the functions build in it.

RECAP OF MID1 AND MID2

MID 1 PRESENTATION

- A road map and project plan was made for the project.
- The tools and IDE to be used was decided .
- Finding of Dataset.
- Cloned the project with github

MID 2 PRESENTATION

- Found a data set.
- Dataset was too large so memory optimization was done.
- CTR Analysis was done based on four features

RESULTS OF THE WORK DONE

App Related Metrics

[206]:
```
app_features = ['app_id', 'app_domain', 'app_category']
```

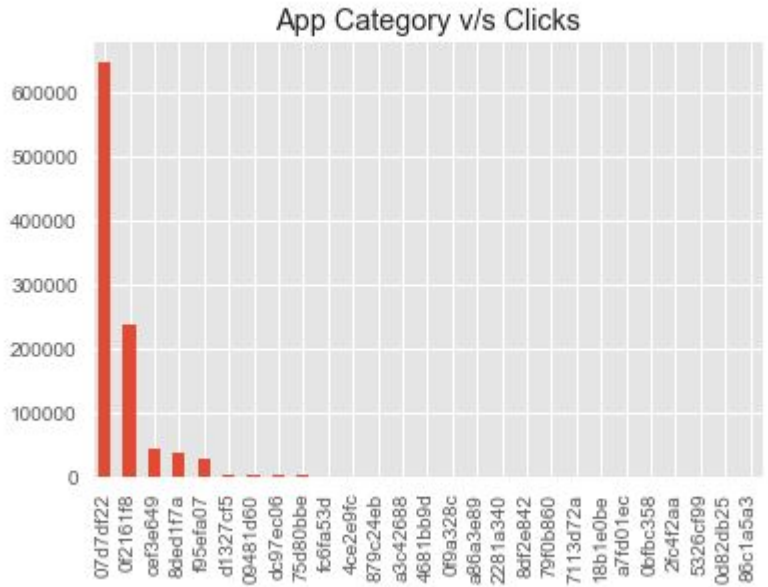
[207]:
```
train_data.groupby('app_category').agg({'click':'sum'}).sort_values(by='click',ascending = False)
```

a3c42688	14.0
a86a3e89	11.0
8df2e842	7.0
79f0b860	1.0
7113d72a	1.0
18b1e0be	1.0
0bfbc358	1.0
86c1a5a3	0.0
2fc4f2aa	0.0
2281a340	0.0
a7fd01ec	0.0
5326cf99	0.0
0d82db25	0.0

[208]:
```
train_data['app_category'].value_counts().plot(kind='bar', title='App Category v/s Clicks')
```

Ouput

```
Out[208]: <matplotlib.axes._subplots.AxesSubplot at 0x20b1fa3eef0>
```

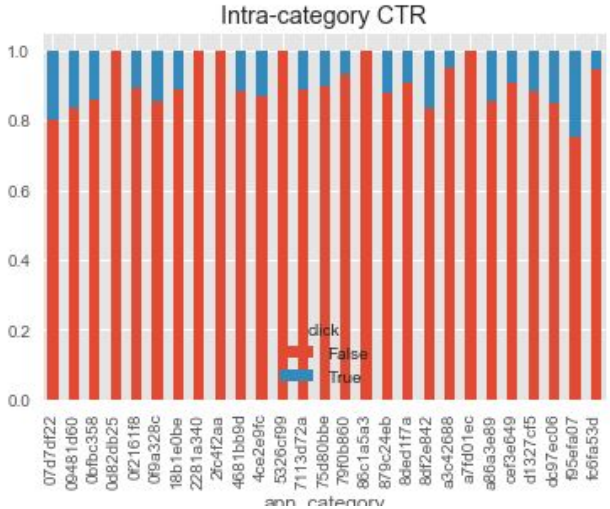


Studying Click Behaviour of app categories

In [211]:
```
train_app_category = train_data.groupby(['app_category', 'click']).size().unstack()
```

In [212]:
```
train_app_category.div(train_app_category.sum(axis=1), axis=0).plot(kind='bar',
    stacked=True, title="Intra-category CTR")
```

Out[212]: <matplotlib.axes._subplots.AxesSubplot at 0x20b22ddb5f8>



DEVELOPING PREDICTION MODEL

Stage 1:Data Preparation Stage

In [314]:
```
model_features = ['weekday', 'hour_in_day',
                  'banner_pos', 'site_category',
                  'device_conn_type', 'app_category',
                  'device_type']
```

In [315]:
```
model_target = 'click'
```

In [ ]:
```
train_model = train_data[model_features+[model_target]].sample(frac=0.1,random_state=42)
```

In order to speeden up the computation,the model features are clubbed with the target features and 10% data is selected for training purpose.

```
In [ ]: train_model = one_hot_features(train_model,
                                       ['site_category',
                                        'app_category',
                                        'banner_pos'])
```

- Features site_category and app_category are hashed into readable format.
- Banner_pos is represented as integers and hence One hot encoding is used to deal with those features.

One hot encoding

	id	click	hour	C1	banner_pos	site_id	site_domain	site_category	app_id	app_domain	...	C15	C16	C17	C18	C19	C2
0	10010966574628106108	True	2014-10-21	1005	0	85f751fd	c4e18dd6	50e219e0	0acbeaa3	45a51db4	...	320	50	2161	0	35	3449
1	10018563981679953217	False	2014-10-21	1005	0	85f751fd	c4e18dd6	50e219e0	8bfb92e0	7801e8d9	...	320	50	1996	1	41	3452
2	10030228488972929850	False	2014-10-21	1005	0	1fbe01fe	f3845767	28905ebd	ecad2386	7801e8d9	...	320	50	1722	0	35	3454
3	10031998322520623865	False	2014-10-21	1005	0	6c5b482c	7687a86e	3e814130	ecad2386	7801e8d9	...	300	250	1994	2	39	3454
4	10032235721168274495	False	2014-10-21	1005	0	1fbe01fe	f3845767	28905ebd	ecad2386	7801e8d9	...	320	50	1722	0	35	3454

Feature Selection

- To reduce the dimensional space occupied and to deal with overfitting,cross validation and regularization is used to obtain a trade-off between number of features and F1 score.
- F1 score used as a performance metric because it represents the harmonic mean between precision and recall

```
4]: from sklearn.model_selection import StratifiedKFold
    from sklearn.model_selection import GridSearchCV

    from sklearn.linear_model import LogisticRegression
    from sklearn.feature_selection import SelectFromModel
    from sklearn.metrics import f1_score
```

```
In [285]: num_splits = 3
          c_values = np.logspace(-3,0,7)
```

```
In [286]: stratified_k_fold = StratifiedKFold(n_splits=num_splits)

          scores = np.zeros(7)
          nr_params = np.zeros(7)
```

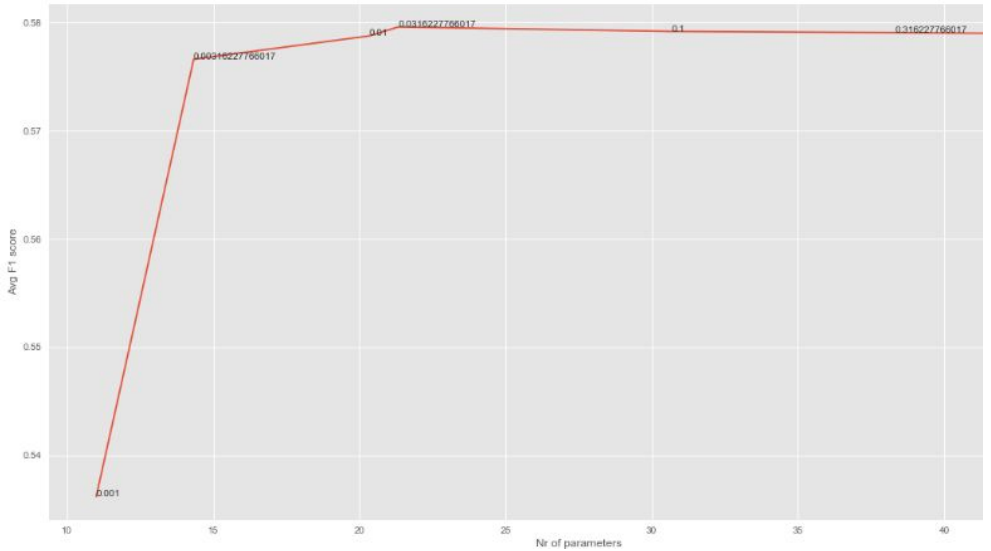
MODEL

Our model is based on Logistic regression and L1 Regularization with balanced weights.

In [ ]:
```
for train_data, valid_data in stratified_k_fold.split(x_train,
                                                      y_train):
    for i, c in enumerate(np.logspace(-3, 0, 7)):
        lr_classify = LogisticRegression(penalty='l1',
                                         class_weight='balanced',
                                         C = c)
        lr_classify.fit(x_train[train_data],
                        y_train[train_data])

        #validation_Set evaluation

        y_prediction = lr_classify.predict(x_train[valid_data])
        score_f1 = f1_score(y_train[valid_data],
                            y_prediction, average='weighted' )

        scores[i] += score_f1 / num_splits

        ### spot the selected parameters ##

        model_selected = SelectFromModel(lr_classify, prefit=True)
        nr_params[i] += np.sum(model_selected.get_support()) / num_splits
```

In [254]:
```
plt.figure(figsize=(20, 10))
plt.plot(nr_params, scores)
```

```
<matplotlib.text.Text at 0x20b2b047320>
```



Parameters obtained using c = 0.1 manage to reduce parameters dimension which optimizes the execution time also improving generalization capacity.

In [288]:
```
lr_classify = LogisticRegression(C=0.1, class_weight='balanced', dual=False,
         fit_intercept=True, intercept_scaling=1, max_iter=100,
         multi_class='ovr', n_jobs=1, penalty='l1', random_state=None,
         solver='liblinear', tol=0.0001, verbose=0, warm_start=False)
```

In [289]:
```
lr_classify.fit(x_train, y_train)
```

Out[289]:
```
LogisticRegression(C=0.1, class_weight='balanced', dual=False,
         fit_intercept=True, intercept_scaling=1, max_iter=100,
         multi_class='ovr', n_jobs=1, penalty='l1', random_state=None,
         solver='liblinear', tol=0.0001, verbose=0, warm_start=False)
```

In [290]:
```
model_selected = SelectFromModel(lr_classify,
                                 prefit=True )
```

In [292]:
```
#pruned_params = model_selected.get_support()
pruned_params
```

Out[292]:
```
array([ True,  True,  True,  True,  True, False,  True,  True,  True,
       False,  True, False, False,  True, False, False,  True,  True,
       False, False, False, False,  True, False,  True, False,  True,
        True,  True,  True, False, False,  True, False, False, False,
       False, False,  True, False, False,  True, False, False, False,
        True, False,  True, False, False,  True,  True, False,  True,
        True,  True, False,  True, False, False,  True,  True], dtype=bool)
```

In [293]:
```
model_features = model_features[pruned_params]
```

GRADIENT BOOSTING OF MODEL

```
import xgboost
from xgboost import XGBClassifier
from sklearn.metrics import classification_report
```

```
x_train, x_valid, y_train, y_valid = train_test_split(
    x_train,
    y_train,
    stratify=y_train,
    test_size=0.1,
    random_state=42)
```

```
model = XGBClassifier()
xgb_clf = model
```

```
In [297]: xgb_clf.fit(x_train, y_train, early_stopping_rounds=10,
                      eval_metric="logloss", eval_set=[(x_valid, y_valid)])
[43]    validation_0-logloss:0.439405
[44]    validation_0-logloss:0.439335
[45]    validation_0-logloss:0.4393
[46]    validation_0-logloss:0.439268
[47]    validation_0-logloss:0.439116
[48]    validation_0-logloss:0.439072
[49]    validation_0-logloss:0.439018
[50]    validation_0-logloss:0.438988
[51]    validation_0-logloss:0.438959
[52]    validation_0-logloss:0.438919
[53]    validation_0-logloss:0.438886
[54]    validation_0-logloss:0.438859
[55]    validation_0-logloss:0.438836
[56]    validation_0-logloss:0.438813
[57]    validation_0-logloss:0.438782
[58]    validation_0-logloss:0.438764
[59]    validation_0-logloss:0.438691
[60]    validation_0-logloss:0.43867
[61]    validation_0-logloss:0.438655
[62]    validation_0-logloss:0.438531
```

```
In [298]: y_pred = xgb_clf.predict(x_test)
          predictions = [round(value) for value in y_pred]
```

```
In [301]: print(classification_report(y_test,
                                      predictions))
```

Log Loss values measuring the performances of a classification models where the prediction label is a value between 0 and 1. The goal of the model is to minmize this value

PERFORMANCE METRICS

```
y_pred = xgb_clf.predict(x_test)
predictions = [round(value) for value in y_pred]
```

```
print(classification_report(y_test,
                            predictions))
```

	precision	recall	f1-score	support
False	0.83	1.00	0.91	248673
True	0.68	0.00	0.00	51328
avg / total	0.80	0.83	0.75	300001

```
from sklearn import metrics

print(metrics.accuracy_score(y_test, predictions))
print(metrics.confusion_matrix(y_test, predictions))
print(metrics.roc_auc_score(y_test, predictions))
```

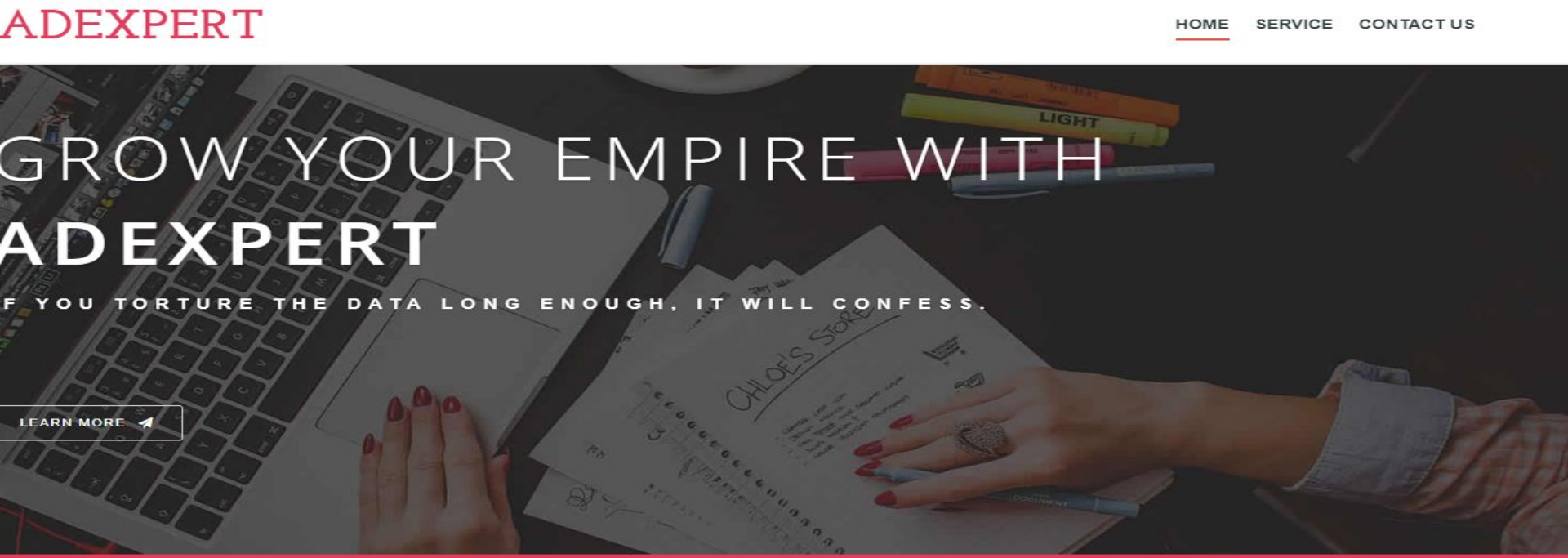
```
0.829060569798
[[248633     40]
 [ 51242     86]]
0.500757322471
```

The model has an 83% accuracy score and 0.5 is the area under the receiver operating characteristic curve.

DIFFICULTIES FACED

- Finding Dataset
- Processing huge Dataset over 6GB
- High end GPU required for training model
- Could not Integrate kernel with github for parallel work
- Less features were available in the previous dataset for the use of Clustering algorithm

FINAL DELIVERABLE



Our Service

Our analysis includes the activities to help organisation make strategic decisions, achieve major goals and solve complex problems, by collecting, analyzing and reporting the most useful information relevant to organisation's needs.

Steps

1.Insert your dataset as a CSV file.
2.Click on start analysis.
3.Sit back and relax your strategies are building.
.

Enter your Dataset here

Choose File No file chosen

Just one click away

Hey! What are you waiting for?

Start Analysis

THANK YOU