- Last Name: Narang
- Date : 6/10/2020
- Approach : Reading data >> EDA >> missing value imputation and Feature engineering >> Model Experimentation and final model selection >> prediction on Test data
- Estimated AUC: 0.85 (based on cross validation mean score)

import required libraries

In [2]:
```
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:100% !important; }</style>"))
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import pandas_profiling
from pandas_profiling import ProfileReport
import matplotlib.pyplot as plt

import numpy as np
import os
import seaborn as sns
import pickle


from sklearn.metrics import roc_auc_score
from sklearn.preprocessing import MinMaxScaler, StandardScaler


from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import learning_curve
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
import graphviz
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RandomizedSearchCV
```

Reading data files

In [0]:
```
train_df = pd.read_csv('/content/train.csv')
test_df = pd.read_csv('/content/test.csv')
```

In [4]:
```
print(train_df.shape)
print(test_df.shape)
```

```
(10000, 10)
(10000, 9)
```

Exploratory Data Analysis

In [5]:
```
profile_train = ProfileReport(train_df)
profile_train
```

Overview

Dataset statistics

Number of variables	10
Number of observations	10000
Missing cells	269
Missing cells (%)	0.3%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	781.4 KiB
Average record size in memory	80.0 B

Variable types

NUM	4
CAT	4
BOOL	2

Reproduction

Analysis started	2020-06-10 15:30:18.656961
Analysis finished	2020-06-10 15:30:26.647893

Out[5]:

In [6]:
```
profile_test = ProfileReport(test_df)
profile_test
```

Overview

Dataset statistics

Number of variables	9
Number of observations	10000
Missing cells	249
Missing cells (%)	0.3%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	703.2 KiB
Average record size in memory	72.0 B

Variable types

NUM	4
CAT	4
BOOL	1

Reproduction

Analysis started	2020-06-10 15:30:30.962351
Analysis finished	2020-06-10 15:30:35.600473

Out[6]:

Feature Engineering and missing Value imputation

In [0]:
```
train_df.gender.fillna('other',inplace=True)
```

In [0]:
```
test_df.gender.fillna('other',inplace=True)
```

In [0]:
```
# categorical variable to one hot encoding
train_df = pd.get_dummies(train_df,drop_first = True)
```

In [10]:
```
train_df.head()
```

Out[10]:

	age	cost_of_ad	in_initial_launch_location	income	n_drivers	n_vehicles	prior_ins_tenure	out
0	56	0.005737	0	62717	2	1	4	
1	50	0.004733	0	64328	2	3	2	
2	54	0.004129	0	83439	1	3	7	
3	16	0.005117	0	30110	2	3	0	
4	37	0.003635	0	76565	2	1	5	

In [0]:
```
test_df = pd.get_dummies(test_df,drop_first = True)
```

In [12]:
```
test_df.head()
```

Out[12]:

	age	cost_of_ad	in_initial_launch_location	income	n_drivers	n_vehicles	prior_ins_tenure	dev
0	34	0.005134	1	40376	1	3	7	
1	53	0.005223	1	84511	1	1	11	
2	46	0.004939	0	79322	1	1	4	
3	36	0.004924	0	63295	1	2	0	
4	28	0.005146	1	36170	1	3	3	

Model - Data Prepration

In [0]:
```
X_train= train_df.drop('outcome',axis = 1)
y_train = train_df.outcome
```

In [0]:
```
#Scaling data from 0 to 1
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
test_df = scaler.transform(test_df)
```

In [15]:
```
X_train.shape,test_df.shape
```

Out[15]:
```
((10000, 13), (10000, 13))
```

Model training and testing

In [0]:
```
from numpy import mean
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from xgboost import XGBClassifier
```

In [17]:
```
# estimate scale_pos_weight value
estimate = y_train.value_counts()[0]/y_train.value_counts()[1]
print('Estimate: %.3f' % estimate)
```

```
Estimate: 9.183
```

In [0]:
```
model = XGBClassifier(random_state=1,)#scale_pos_weight = estimate
```

In [19]:
```
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
scores = cross_val_score(model, X_train, y_train, scoring='roc_auc', cv=
cv, n_jobs=-1)
print('Mean ROC AUC: %.3f' % mean(scores))
```

```
Mean ROC AUC: 0.850
```

In [23]:
```
# tuning model
# define grid
param_grid  = {"learning_rate"    : [0.05, 0.10, 0.15 ] ,
               "max_depth"        : [ 3, 4, 5, 6, 8],
               "min_child_weight" : [ 1, 3, 5 ],
               "gamma"            : [ 0.0, 0.1, 0.2 , 0.3],
               "colsample_bytree" : [ 0.3, 0.4, 0.5 ] ,
               "scale_pos_weight":  [1,3,6]}

# define evaluation procedure
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)

# define grid search
grid = RandomizedSearchCV(estimator=model, param_distributions=param_gri
d, n_jobs=-1, cv=cv, scoring='roc_auc')

# execute the grid search
grid_result = grid.fit(X_train, y_train)

# report the best configuration
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_p
arams_))
```

```
Best: 0.849953 using {'scale_pos_weight': 1, 'min_child_weight': 5, 'ma
x_depth': 3, 'learning_rate': 0.15, 'gamma': 0.1, 'colsample_bytree':
0.4}
```

test data predition

In [0]:
```
#Best model
model = XGBClassifier(learning_rate =0.15 ,max_depth = 3,min_child_weigh
t = 5,
                      gamma =0.1 ,colsample_bytree =0.4 ,scale_pos_weigh
t = 1)
```

In [25]:
```
model.fit(X_train,y_train)
```

Out[25]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.4, gamma=0.1,
              learning_rate=0.15, max_delta_step=0, max_depth=3,
              min_child_weight=5, missing=None, n_estimators=100, n_job
s=1,
              nthread=None, objective='binary:logistic', random_state=
0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

In [0]:
```
pred_test = model.predict(test_df)
```

In [0]:
```
df_out = pd.read_csv('/content/test.csv')
```

In [0]:
```
df_out['pred_outcome'] = pred_test
```

In [34]:
```
df_out.head()
```

Out[34]:

	age	cost_of_ad	device_type	gender	in_initial_launch_location	income	n_drivers	n_vehicles
0	34	0.005134	Android	F	1	40376	1	3
1	53	0.005223	desktop	F	1	84511	1	1
2	46	0.004939	laptop	F	0	79322	1	1
3	36	0.004924	Android	F	0	63295	1	2
4	28	0.005146	other	F	1	36170	1	3

In [133]:
```
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
scores = cross_val_score(model, X_train, y_train, scoring='roc_auc', cv=
cv, n_jobs=-1)
# summarize performance
print('Mean ROC AUC: %.5f' % mean(scores))
```

```
Mean ROC AUC: 0.84996
```