Snapchat Political Ads

- **See the main project notebook for instructions to be sure you satisfy the rubric!**
- See Project 03 for information on the dataset.
- A few example prediction questions to pursue are listed below. However, don't limit yourself to them!
    - Predict the reach (number of views) of an ad.
    - Predict how much was spent on an ad.
    - Predict the target group of an ad. (For example, predict the target gender.)
    - Predict the (type of) organization/advertiser behind an ad.

Be careful to justify what information you would know at the "time of prediction" and train your model using only those features.

Summary of Findings

Introduction

We use 2 features in the baseline model, which I thought will well predict the impression for ads. One is Spend and one is Gender column. The Spend column is quantitative column and the Gender column is We think these two columns are correlated with the Impression. For the nominal column. the performance of the model, we used onehotencoder to deal with categorical column and linear regression model to predict the quantitative number. Also, we use the SimpleImputer to fill the none value in the dataset. We use the train_test_split function to divided the test and train data. We use the cross-validation method (divided in 2 groups) to check the variance of the model and we found out the score has a quiet large difference, which shows the variance is high. Furthermore, we can notice that the score for this model is 0.448119760102611. We use the for loop and run this model for 100 times and record it score. As we can notice in this graph the mean score of this model is 0.4346286067229543. All of the scores determined the baseline model is not a good model to predict the impression of ads for this dataset. We will fix the baseline model in the final model part.

Baseline Model

We use 2 features in the baseline model, which I thought will well predict the impression for ads. One is Spend and one is Gender column. The Spend column is quantitative column and the Gender column is categorical column. We think these two columns are correlated with the Impression. For the categorical column, the performance of the model, we used onehotencoder to deal with categorical column and we use the linear regression model to predict the quantitative number. Also, we use the Simple Imputer to fill the none value in the dataset in numerical column. At last, we use the column transformer to put categorical and numerical into the pipeline and use linear regression to predict it. We use the train_test_split function to divided the test and train data and use the cross-validation method (divided in 2 groups) to check the variance of the model and we found out the score has a quiet large difference, which shows the variance is high. Furthermore, we can notice that the score for this model is 0.448119760102611. We use the for loop and run this model for 100 times and record it score. As we can notice in this graph the mean score of this model is 0.4346286067229543. All of the scores determined the baseline model is not a good model to predict the impression of ads for this dataset. We will fix the baseline model in the final model part.

Final Model

For this model, we add 3 new features to improve out baseline model. Also, we transform the start time and end time to time elapse. The first feature is age period. For this one, we divided age into 4 different peirods. The helper function is to change the "all age" to "all", the age which are end with "+" and "-" to '+' and '-', and the other age into 'between'. The second feature is checking the person is weather adult or not. We divide the age into 2 regions according to whether they are older than 18 or not. The third one is about the gender. we change the gender in to 1,2,0 according to Female, Male, and all. The reason why we think it is good for our data is that:

1. By adding those columns in to our model, this can let the model more fit on our data according to different classes that we made for those features we add.
2. The features we used before are not so good, as we can notice that the score for our previous model is pretty low, so we need to change that
3. we can find that there are some more data that can contribute to our model and let the model fit more to the data. We can predict our target (impressions) more accurate.
4. We divided the target population to adult or non-adult people, which is categorical columns. Also, we set the age range for them. Those classifications help us to determined what types of population will get a higher impression or not.
5. The timeclip(numerical columns), we added is also useful, since if the ads post for long time, it might has more impressions than shorter one.
6. The gender(categorical) of target population also take a role in it. The specified gender seems to have less impression than targeting all gender.

The model we choose regression model, for the parameters that ended up performing best, the method of model selection used.

1. Since we are predicting the number of impression of ads, we are using the regression model to predict
2. We use the functiontransformer to use the helper function and we use the columntransformer to let it easily fit the training data and transform the rest data and

change the features into a more efficient way from the giving feature.

3. we use the PCA in the pipeline, which can drop the correlated column after the onehotcoder for categorical columns and this can increase our score of model.
4. We use SimpleImputer to completing missing values. Also, we use StandardScaler to standardize features by removing the mean and scaling to unit variance.
5. We use the OneHotEncoder to encode categorical features as a one-hot numeric array and use PCA to drop the correlated columns.
6. Finally, we use LinearRegression to predict our data.

for the features: timeclip we add and why this feature is good for our model, there are following reasons:

1. For getting the time gap between the starttime and the endtime, we think different time period can influence the amount of impressions.
2. fit the model according to the influence of different time gap on impressions

We split the data into test and train dataset. And use the cross classification(divided into two groups). We got [0.47444666, 0.55295702], which is much close value than the baseline model. Then we test the score for the test data into the improved model. We got 0.7977883833069321. It is a pretty high score. It much higher than the baseline model. We take a for loop and takes 100 times to calculate the score of model. Now we can notice that out mean model's score become 0.5308069737837031. It also higher than the mean of baseline

Fairness Evaluation

The interesting subset of the data would be the gender column. There are three types of the columns. There are target only to Male, only to Female and to all gender people. I am interested in two groups the specified gender target and to all gender target. I checked the bias of this by using the permutation test and use the rmse to test the 'fairness'. Since we made a better final model previously. I use the final model to predit the impression of each ads. Then, I divide my data into two group by targeting specified gender and targeting all gender. ThenmI find the rmse method to measure the observation for the whole data.

Then I used the permutation test to check the specified gender target group and shuffled the perdiction of it. I decided to choose the significant level of 0.01. The null hypothesis is the the distribution of impression between specified gender target group and the whole data are the same. The alternative hypothesis is the the distribution of impression between specified gender target group and the whole data are not the same. I shuffled the new column which target to specified, and use a for loop for 100 times. Then I found out the p-value of 0.57, which means we fail to reject the null hypothesis.This means the data is not bias between the specific gender with the whole data group on Impression.

Then I used the permutation test to check the all gender target group and shuffled the perdiction of it. I decided to choose the significant level of 0.01. The null hypothesis is the the distribution of impression between all gender target group and the whole data are the same. The alternative hypothesis is the distribution of impression between all gender target group and the whole data are not the same. I shuffled the new column which target to all gender type, and use a for loop for 100 times. Then I found out the p-value of 0.0, which means we reject the null hypothesis. This means the data is bias on the all gender type group comparing with the whole data group on Impression.

As the result, the ads target to all gender group has bias, but the ads target toward the specified gender is not bias comparing the whole dataset.

Code

In [3]:
```
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
import seaborn as sns
%matplotlib inline
%config InlineBackend.figure_format = 'retina'  # Higher resolution figu
from sklearn.preprocessing import FunctionTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import Binarizer
from sklearn.preprocessing import MinMaxScaler
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression

from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import Imputer
```

In [4]:
```
# combined data
fp = os.path.join('data', 'PoliticalAds2018.csv')
data2018 = pd.read_csv(fp)
fp2 = os.path.join('data', 'PoliticalAds2019.csv')
data2019 = pd.read_csv(fp2)
list1 = [data2018,data2019]
data2019.shape[0]+data2018.shape[0]
combined_data = pd.concat(list1,ignore_index = True)
pd.set_option('display.max_columns', None)
combined_data.head()
```

Out[4]:

	ADID	CreativeUrl	Spend	Imp
0	91db2796a80472ed8c2bfa17760b3ce1471f6ec1f3147b...	https://www.snap.com/political-ads/asset/b2c47...	1044	
1	97e3f17d5ec164c454a35d2822734482ca60be3f3af310...	https://www.snap.com/political-ads/asset/affc7...	279	
2	14535fea019a9b1a910a77ce1555af8bdedbb5c78fb60a...	https://www.snap.com/political-ads/asset/754f6...	6743	
3	10b64550ad4a23c651d7883746cabeac93cbd92d5f3b3f...	https://www.snap.com/political-ads/asset/818ae...	3698	
4	2438786c60ae41cf56614885b415a72857bbfb5c06f760...	https://www.snap.com/political-ads/asset/2c264...	445	

In [5]:
```
# cleaning data
combined_data['Gender'] = combined_data['Gender'].astype(str).apply(lamb
combined_data['StartDate'] = pd.to_datetime(combined_data['StartDate'])
combined_data['StartDate'] = combined_data['StartDate'].dt.tz_convert('U
combined_data['EndDate'] = pd.to_datetime(combined_data['EndDate'])
combined_data['EndDate'] = combined_data['EndDate'].dt.tz_convert('US/Ce
nuique_languages = combined_data['Language'][combined_data['Language'].i
convert_languages = {'en':'English', 'de': 'German', 'sv':'Swedish', 'nl
       'fr':'French', 'nb':'Norwegian Bokmål', 'fi':'Finnish', 'nl,en':'
combined_data['Language'] = combined_data['Language'][combined_data['Lan
```

In [6]:
```
combined_data.head()
```

Out[6]:

	ADID	CreativeUrl	Spend	Imp
0	91db2796a80472ed8c2bfa17760b3ce1471f6ec1f3147b...	https://www.snap.com/political-ads/asset/b2c47...	1044	
1	97e3f17d5ec164c454a35d2822734482ca60be3f3af310...	https://www.snap.com/political-ads/asset/affc7...	279	
2	14535fea019a9b1a910a77ce1555af8bdedbb5c78fb60a...	https://www.snap.com/political-ads/asset/754f6...	6743	
3	10b64550ad4a23c651d7883746cabeac93cbd92d5f3b3f...	https://www.snap.com/political-ads/asset/818ae...	3698	
4	2438786c60ae41cf56614885b415a72857bbfb5c06f760...	https://www.snap.com/political-ads/asset/2c264...	445	

Baseline Model

In [158]:
```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import FunctionTransformer
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.decomposition import PCA
from sklearn.model_selection import cross_val_score
```

In [8]:
```
# This two columns will not fit in encoder
data = combined_data.drop(['StartDate','EndDate'],axis = 1)
```

In [236]:
```
# deal with catagorical column and numerical col and predit the result
y = data['Impressions']
X = data.drop('Impressions', axis=1)
catcols = ['Gender']
numcols = ['Spend']

cats = Pipeline([
    ('imp', SimpleImputer(strategy='constant', fill_value='NULL')),
    ('ohe', OneHotEncoder(handle_unknown='ignore', sparse=False)),
   ])

ct = ColumnTransformer([
    ('catcols', cats, catcols),
    ('numcols', SimpleImputer(strategy='constant', fill_value=0), numcol

pl = Pipeline([('feats', ct), ('reg', LinearRegression())])
```

In [255]:
```
X_tr, X_ts, y_tr, y_ts = train_test_split(X, y, test_size=0.2)
scores = cross_val_score(pl, X_tr, y_tr, cv=2)
scores
```

Out[255]:
```
array([0.44471712, 0.77808266])
```

In [256]:
```
#X_tr, X_ts, y_tr, y_ts = train_test_split(X, y, test_size=0.3)
pl.fit(X_tr, y_tr)
pl.score(X_ts, y_ts)
```

Out[256]:
```
0.448119760102611
```

In [395]:
```
out = []
for _ in range(100):
    X_tr, X_ts, y_tr, y_ts = train_test_split(X, y, test_size=0.10)
    pl.fit(X_tr, y_tr)
    out.append(pl.score(X_ts, y_ts))
```

In [396]:
```
np.mean(out)
```

Out[396]:
```
0.4346286067229543
```

Final Model

In [16]:
```
combined_data.head()
```

Out[16]:

	ADID	CreativeUrl	Spend	Impre
0	91db2796a80472ed8c2bfa17760b3ce1471f6ec1f3147b...	https://www.snap.com/political-ads/asset/b2c47...	1044	
1	97e3f17d5ec164c454a35d2822734482ca60be3f3af310...	https://www.snap.com/political-ads/asset/affc7...	279	
2	14535fea019a9b1a910a77ce1555af8bdedbb5c78fb60a...	https://www.snap.com/political-ads/asset/754f6...	6743	3
3	10b64550ad4a23c651d7883746cabeac93cbd92d5f3b3f...	https://www.snap.com/political-ads/asset/818ae...	3698	
4	2438786c60ae41cf56614885b415a72857bbfb5c06f760...	https://www.snap.com/political-ads/asset/2c264...	445	

In [17]:
```
# add the time elapse
combined_data['StartDate'] = pd.to_datetime(combined_data['StartDate'])
combined_data['StartDate'] = combined_data['StartDate'].dt.tz_convert('U
combined_data['EndDate'] = pd.to_datetime(combined_data['EndDate'])
combined_data['EndDate'] = combined_data['EndDate'].dt.tz_convert('US/Ce
combined_data['timeclip'] = combined_data.EndDate - combined_data.StartD
```

In [18]:
```
combined_data['timeclip'] = combined_data['timeclip'].apply(lambda x: x.
```

In [24]:
```
combined_data['AgeBracket'] = combined_data['AgeBracket'].astype(str).re
```

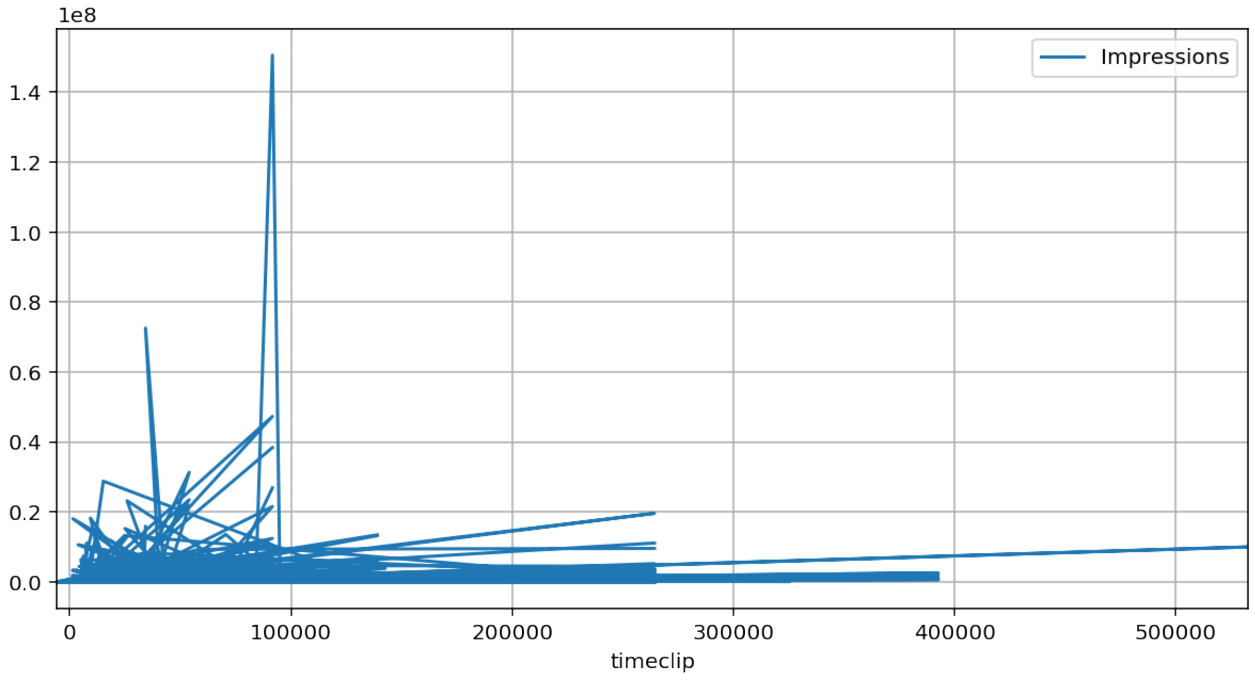
In [297]:
```
combined_data['AgeBracket1'] = combined_data['AgeBracket']
```

In [397]:
```
import numpy as np
import pandas as pd
graphusing = combined_data.copy()
d = {'one' : graphusing['Impressions'],
     'two' : graphusing['timeclip']}

df = pd.DataFrame(d)

graphusing.plot(x='timeclip', y=['Impressions'], figsize=(10,5), grid=Tr
```

Out[397]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1d0172e8>
```



In [298]:
```
def process_age(agebracket):
    lst = []
    idx = agebracket.index.tolist()
    for age in agebracket.values:
        num = age[0]
        if num =='ALL':
            lst.append('all')
        elif num[-1]=='+':
            lst.append('+')
        elif num[-1]=='-':
            lst.append('-')
        else:
            lst.append('between')

    return pd.DataFrame(lst, index = idx)
```

In [299]:
```
def genderchange(gender):
    lst2 = []
    idx = gender.index.tolist()
    for age in gender.values:
        num = age[0]
        if num == 'FEMALE':
            lst2.append(1)
        elif num == 'MALE':
            lst2.append(2)
        else:
            lst2.append(0)
    return pd.DataFrame(lst2, index = idx)
```

In [300]:
```
def agerange(agebracket):
    lst1 = []
    idx = agebracket.index.tolist()
    for age in agebracket.values:
        num = age[0]
        if num>'18':
            lst1.append(1)
        else:
            lst1.append(0)
    return pd.DataFrame(lst1, index = idx)
```

In [338]:
```
data_drop_useless = data_drop_useless.copy()
X = data_drop_useless.drop(['Impressions','StartDate','EndDate'], axis =
y = data_drop_useless.Impressions

num_col = ['Spend','timeclip']
num = Pipeline([
    ('null',SimpleImputer(strategy='constant', fill_value=0)),
    ('std',StandardScaler())
])
#new feature (1st new feature is timeclip)
age_pl = Pipeline([
    ('age',FunctionTransformer(process_age, validate = False)),
    ('ohe',OneHotEncoder(handle_unknown = 'ignore',sparse = False))
])
age_col = ['AgeBracket']

#3rd new feature
agerangecol = ['AgeBracket1']
agerange = Pipeline([
    ('age1',FunctionTransformer(agerange, validate = False)),
    ('ohe1',OneHotEncoder(handle_unknown = 'ignore',sparse = False))
])
#new feature
gendercol = ['Gender']
agerange = Pipeline([
    ('gender',FunctionTransformer(genderchange, validate = False))
])

cal_col = ['Gender']
cal1 = Pipeline([
    ('imp', SimpleImputer(strategy='constant', fill_value='NULL')),
    ('ohe', OneHotEncoder(handle_unknown='ignore', sparse=False)),
    ('pca', PCA(svd_solver='full', n_components=0.99))
])

cats = Pipeline([
    ('imp', SimpleImputer(strategy='constant')),
    ('ohe', OneHotEncoder(handle_unknown='ignore'))
])

processor = ColumnTransformer([
    ('num',num,num_col),
    ('cal1',cal1,cal_col),
    ('age',age_pl,age_col),
    ('agerange',agerange,agerangecol),
    ('genderchange',agerange,gendercol),
    ('numcols', SimpleImputer(strategy='constant'), numcols)
])
process = Pipeline([
    ('num',processor),
    ('linear',LinearRegression())
])
```

In [363]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30
scores = cross_val_score(process, X_train, y_train, cv=2)
scores
```

Out[363]:
```
array([0.47444666, 0.55295702])
```

In [364]:
```
process.fit(X_train,y_train)
process.score(X_test,y_test)
```

Out[364]:
```
0.7977883833069321
```

In [389]:
```
out1 = []
for _ in range(100):
    X_tr, X_ts, y_tr, y_ts = train_test_split(X, y, test_size=0.30)
    process.fit(X_tr, y_tr)
    out1.append(process.score(X_ts, y_ts))
```

In [390]:
```
np.mean(out1)
```

Out[390]:
```
0.5308069737837031
```

In [ ]:

Fairness Evaluation

In [37]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
```

In [55]:
```
combined_data.head()
```

Out[55]:

	ADID	CreativeUrl	Spend	Imp
0	91db2796a80472ed8c2bfa17760b3ce1471f6ec1f3147b...	https://www.snap.com/political-ads/asset/b2c47...	1044	
1	97e3f17d5ec164c454a35d2822734482ca60be3f3af310...	https://www.snap.com/political-ads/asset/affc7...	279	
2	14535fea019a9b1a910a77ce1555af8bdedbb5c78fb60a...	https://www.snap.com/political-ads/asset/754f6...	6743	
3	10b64550ad4a23c651d7883746cabeac93cbd92d5f3b3f...	https://www.snap.com/political-ads/asset/818ae...	3698	
4	2438786c60ae41cf56614885b415a72857bbfb5c06f760...	https://www.snap.com/political-ads/asset/2c264...	445	

In [59]:
```
new = combined_data.copy()
new['predict'] = process.predict(X)
```

In [60]:
```
new['check_gender'] = (new['Gender'].apply(lambda x: False if x == 'All'
new['check_gender'] = new['check_gender'].replace({True:'specified', Fal
```

In [61]:
```
new.head()
```

Out[61]:

nterests	OsType	Segments	LocationType	Language	AdvancedDemographics	Targeting Connection Type	Targeti Carr (IS
NaN	NaN	Provided by Advertiser	NaN	NaN	NaN	NaN	N
Arts & Culture ens,Chat Fiction usiasts...	NaN	Provided by Advertiser	NaN	NaN	NaN	NaN	N
TV Live Event ers (The Academy ards),TV ...	NaN	Provided by Advertiser	NaN	NaN	NaN	NaN	N
NaN	NaN	Provided by Advertiser	NaN	NaN	NaN	NaN	N
NaN	NaN	Provided by Advertiser	NaN	NaN	NaN	NaN	N

In [62]:
```
#specified
spe_data = new[new['check_gender'] == 'specified'][['Impressions','predi
np.sqrt(np.mean((spe_data['predict'] - spe_data['Impressions'])**2))
```

Out[62]:
```
493492.52793122875
```

In [66]:
```
#all
all_data = new[new['check_gender'] == 'all'][['Impressions','predict']]
np.sqrt(np.mean((all_data['predict'] - all_data['Impressions'])**2))
```

Out[66]:
```
2266175.6243760404
```

In [68]:
```
#total
total_data = new[['Impressions','predict']]
obs = np.sqrt(np.mean((total_data['predict'] - total_data['Impressions']
```

In [92]:
```
#permution test
lst = []
for _ in range(100):
    s = (
        new[new['check_gender'] == 'specified']
        .assign(is_spe = new['predict'].sample(frac = 1, replace=False).
    )
    num = np.sqrt(np.mean((s['is_spe'] - s['Impressions'])**2))
    lst.append(num)
```

In [93]:
```
(lst <= obs).mean()
```

Out[93]:
```
0.57
```

In [94]:
```
#permution test
lst1 = []
for _ in range(100):
    s = (
        new[new['check_gender'] == 'all']
        .assign(is_all = new['predict'].sample(frac = 1, replace=False).
    )
    num = np.sqrt(np.mean((s['is_all'] - s['Impressions'])**2))
    lst1.append(num)
```

In [96]:
```
(lst1 <= obs).mean()
```

Out[96]:
```
0.0
```

In [ ]:

In [ ]:

In [ ]: