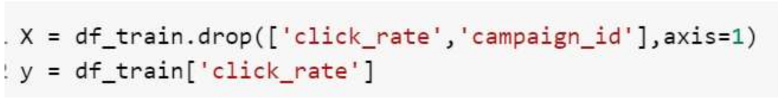
# APPROACH (JOB-A-THON)

Performed **Exploratory Data Analysis** on the test dataset with **Feature Engineering** when required:

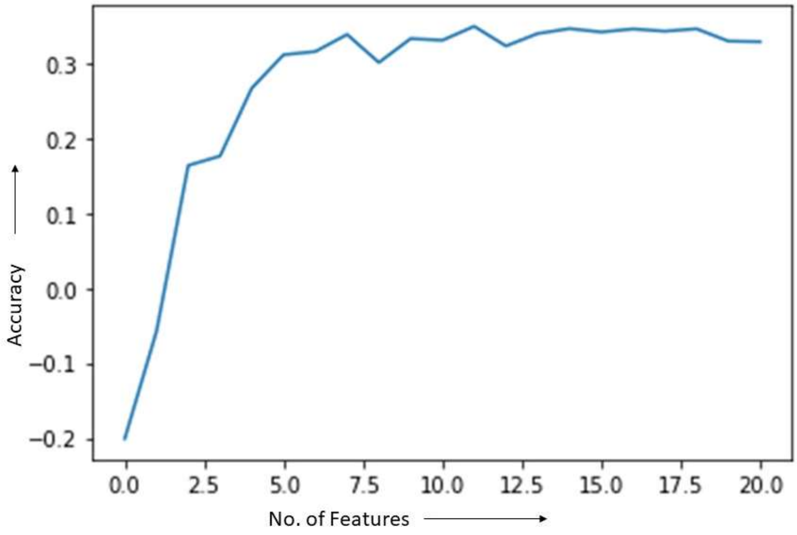
a. The problem at hand was predicting the ‘click_rate’ which is a regression problem
b. Found out that there are no null values present in the dataset
c. Almost all the columns(features) were numerical except for ‘times_of_day’
d. Performed one-hot encoding on the column and appended it to the dataset
e. Dropped the ‘times_of_day’ column from the data

2. Preparation of data for feature selection and training:
   a. It was observed that the ‘campaign_id’ column, although present in the dataset might not have any statistical significance for calculating ‘click_rate’ and hence it was dropped from the independent variable (X). So, X and y:



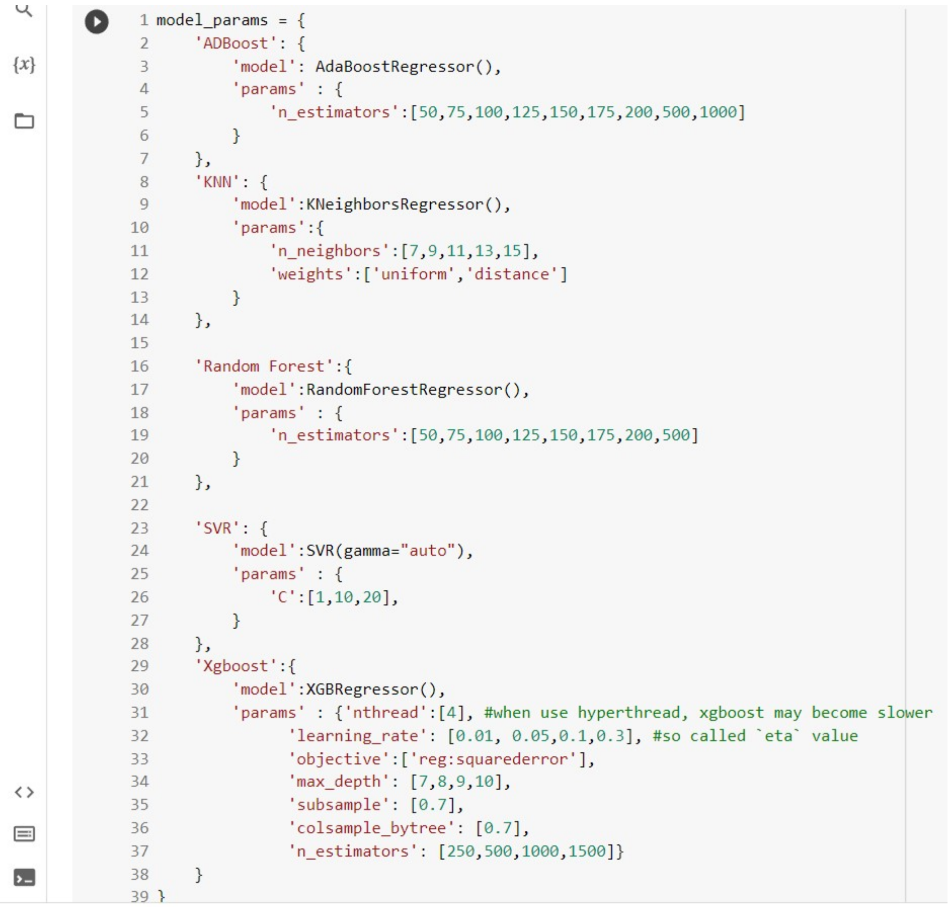
```
X = df_train.drop(['click_rate','campaign_id'],axis=1)
y = df_train['click_rate']
```

   b. From the test set only a validation set was kept aside using sklearn’s train_test_split (X_train, X_val)
3. **Feature Selection** for the model:
   a. At first a base Random Forest model was fit to the X_train.
   b. The feature importance was stored in a list.
   c. A loop was run so as to train another Xgboost model with 1 to all the features in the dataset.
   d. It was found that having 12 most important features gave the best results



4. Grid Search CV **(Model Training):**

   a. The grid search was done over pre-set model parameters:

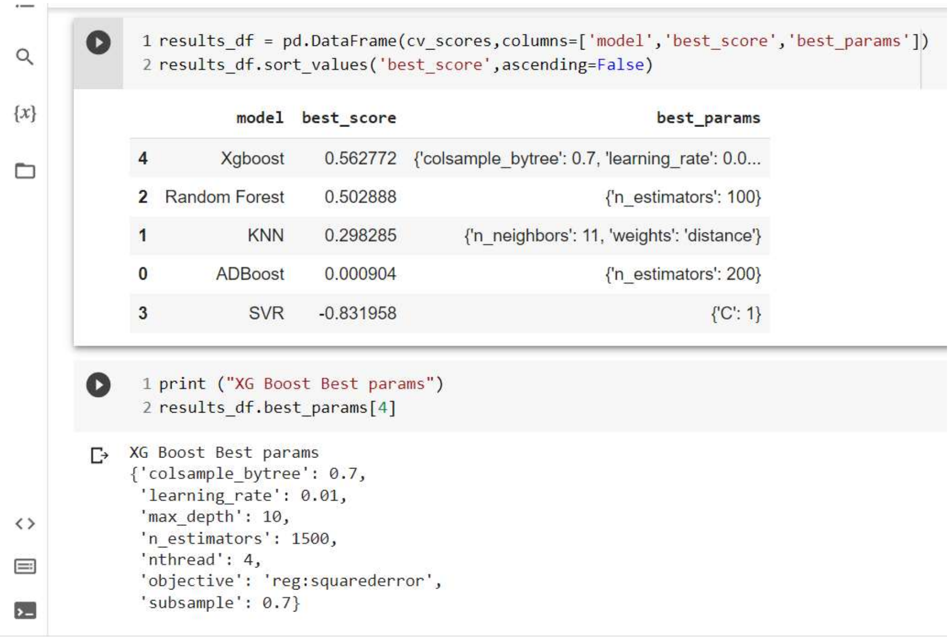


```
model_params = {
    'ADBoost': {
        'model': AdaBoostRegressor(),
        'params' : {
            'n_estimators':[50,75,100,125,150,175,200,500,1000]
        }
    },
    'KNN': {
        'model':KNeighborsRegressor(),
        'params':{
            'n_neighbors':[7,9,11,13,15],
            'weights':['uniform','distance']
        }
    },

    'Random Forest':{
        'model':RandomForestRegressor(),
        'params' : {
            'n_estimators':[50,75,100,125,150,175,200,500]
        }
    },

    'SVR': {
        'model':SVR(gamma="auto"),
        'params' : {
            'C':[1,10,20],
        }
    },
    'Xgboost':{
        'model':XGBRegressor(),
        'params' : {'nthread':[4], #when use hyperthread, xgboost may become slower
               'learning_rate': [0.01, 0.05,0.1,0.3], #so called `eta` value
               'objective':['reg:squarederror'],
               'max_depth': [7,8,9,10],
               'subsample': [0.7],
               'colsample_bytree': [0.7],
               'n_estimators': [250,500,1000,1500]}
    }
}
```

   b. The best params were printed in a dataframe:



```
results_df = pd.DataFrame(cv_scores,columns=['model','best_score','best_params'])
results_df.sort_values('best_score',ascending=False)
```

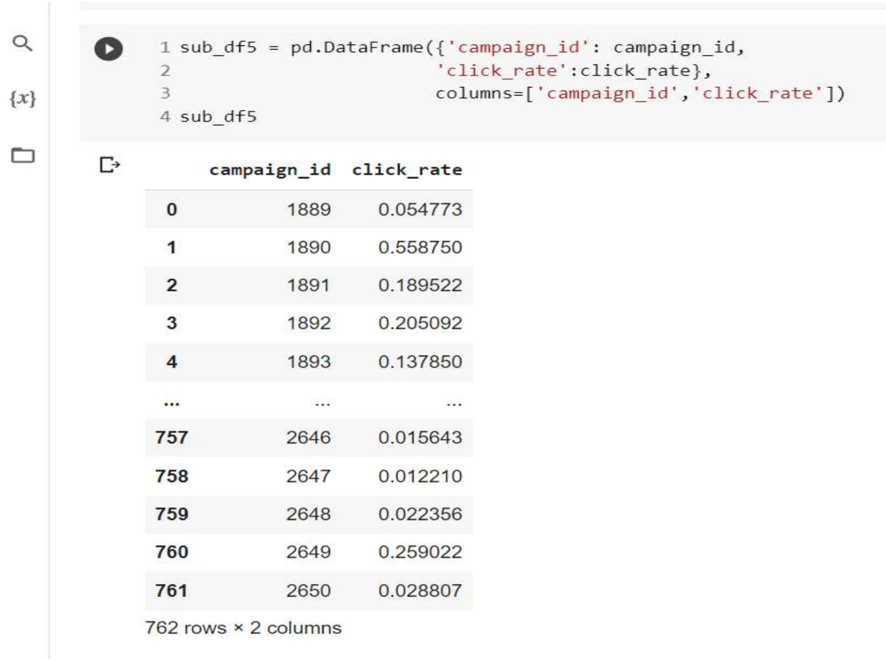
| | model | best_score | best_params |
|---|---|---|---|
| 4 | Xgboost | 0.562772 | {'colsample_bytree': 0.7, 'learning_rate': 0.0... |
| 2 | Random Forest | 0.502888 | {'n_estimators': 100} |
| 1 | KNN | 0.298285 | {'n_neighbors': 11, 'weights': 'distance'} |
| 0 | ADBoost | 0.000904 | {'n_estimators': 200} |
| 3 | SVR | -0.831958 | {'C': 1} |

```
print ("XG Boost Best params")
results_df.best_params[4]
```

```
XG Boost Best params
{'colsample_bytree': 0.7,
 'learning_rate': 0.01,
 'max_depth': 10,
 'n_estimators': 1500,
 'nthread': 4,
 'objective': 'reg:squarederror',
 'subsample': 0.7}
```

5. Test on Validation Set **(Model Evaluation):**
    a. The two best performing models Xgboost and Random forest were trained once again on the train set(X_train) and tested on the validation set(X_val) with the best params as found from grid search.
    b. It was observed that the Random forest model was the best performing model

6. **Test Set** Evaluation1:
    a. The test dataset was prepared like the train data
    b. The RF model was trained using the entire train dataset(X = X_train + X_val)
    c. The final trained model was used to predict the 'click_rate'
    d. The predictions were converted into a dataframe

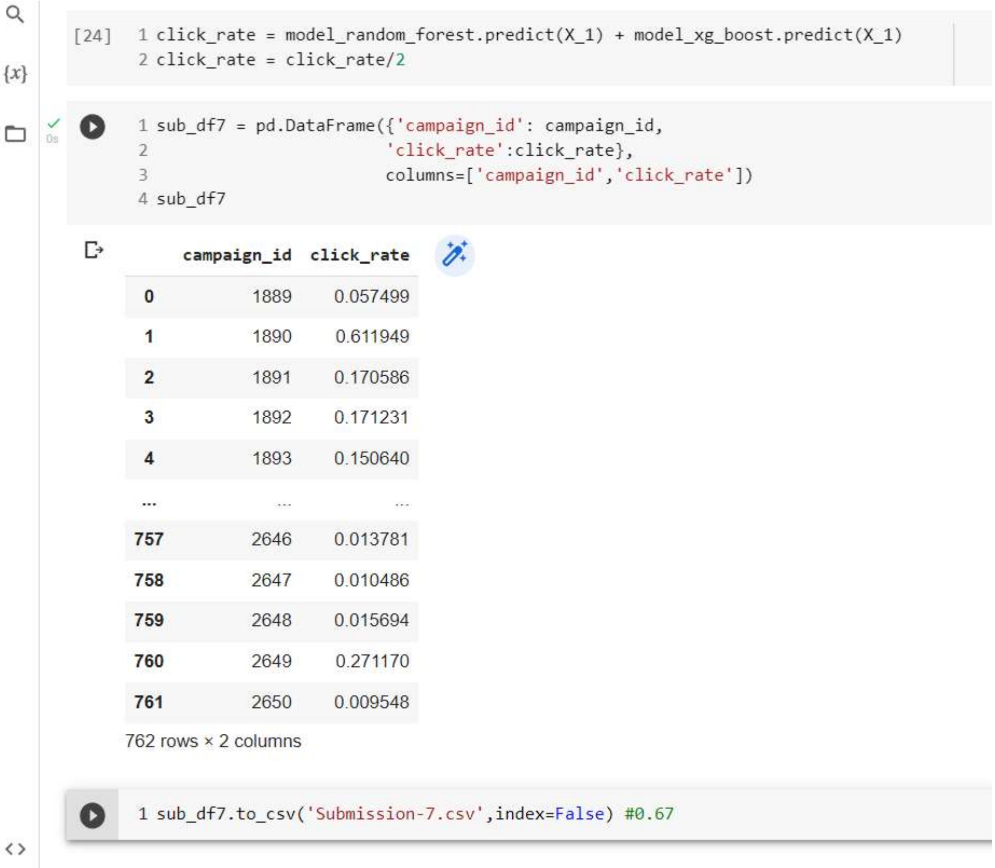


```
sub_df5 = pd.DataFrame({'campaign_id': campaign_id,
                        'click_rate':click_rate},
                        columns=['campaign_id','click_rate'])
sub_df5
```

| | campaign_id | click_rate |
|---|---|---|
| **0** | 1889 | 0.054773 |
| **1** | 1890 | 0.558750 |
| **2** | 1891 | 0.189522 |
| **3** | 1892 | 0.205092 |
| **4** | 1893 | 0.137850 |
| **...** | ... | ... |
| **757** | 2646 | 0.015643 |
| **758** | 2647 | 0.012210 |
| **759** | 2648 | 0.022356 |
| **760** | 2649 | 0.259022 |
| **761** | 2650 | 0.028807 |

762 rows × 2 columns

7. **Test set** Evaluation2(final):
   a. Tried to predict the ‘click_rate’ with both the RF and XGboost model.
   b. Both the models were trained using the entire train dataset (X = X_train + X_val)
   c. The predicted ‘click_rate’ was average of preds of both the models.
   d. The predictions list was converted into a dataframe for submission.



```
click_rate = model_random_forest.predict(X_1) + model_xg_boost.predict(X_1)
click_rate = click_rate/2
```

```
sub_df7 = pd.DataFrame({'campaign_id': campaign_id,
                        'click_rate':click_rate},
                        columns=['campaign_id','click_rate'])
sub_df7
```

```
     campaign_id  click_rate
0           1889    0.057499
1           1890    0.611949
2           1891    0.170586
3           1892    0.171231
4           1893    0.150640
...          ...         ...
757         2646    0.013781
758         2647    0.010486
759         2648    0.015694
760         2649    0.271170
761         2650    0.009548
762 rows × 2 columns
```

```
sub_df7.to_csv('Submission-7.csv',index=False) #0.67
```