Predicting how users with different lifestyle category interest will interact with off-plan multi-lingual Real Estate Video Ads in the GCC* on Snapchat.

Introduction

Motivation

Digital Marketers rely on what is called “Best-Practices” in building their strategies and plans. Throughout my professional experience, I have come across few best practices that became obsolete or that existed earlier due to the limitations when the research was conducted.

Nowadays, I am shifting towards building my own best-practices via digging deep in our own historical performance of the past campaigns while utilizing the knowledge and skills I am gaining from enrollment in the Data Science Msc program.

Especially that with the massive increase in the users targeting capabilities in digital advertising. It became very difficult to use the traditional methods in analyzing campaigns performances for future reference.

ML Challenge & Objective

Hopefully using machine learning in this research, we can build a good machine learning supervised regression model that will be able to predict the performance of paid advertising campaigns on Snapchat Platform for the Off-Plan Real Estate industry by utilizing the ability of easily controlling one of the features “Lifestyle Category” via changing the campaign targeting settings and maybe get some insights to debunk one of the unofficial best practices that are out there.

Data Set - Introduction

The dataset is a 90 days digital marketing campaign report that includes but not limited to features like how many times were the ad served per lifestyle category interest per day, how many interactions were received per lifestyle category per day and with some other features that are concerned with the cost. All campaigns had the same targeting of 21+ for the age, M/F for the gender, Arabic/English for the users language and with the same objective of swipe ups.

Limitations

Due to a global anti-discrimination policy, housing campaigns are only allowed to target the age bracket of 21+, so including age variable will result in redundancy because we can’t apply age targeting anyways in the future.

Since the dataset is for roughly 3 months, the research doesn't look into the annual seasonality factor of the real estate market across the whole year, the research doesn't look into the seasonality factor of the real estate market being affected with covid as well

The research doesn't look into the frequency of ad views per user and how it affects the interaction rate, however in the launched campaigns there was no frequency capping applied on any of them.

Due to the limited marketing activities across other countries in GCC, this research is intended to look at KSA & UAE Performance only.

Limitations that were overcome

The published campaigns were advertising different products with a wide range of price. Since each campaign was advertising a different product, the limitation was overcome by converting the campaign name to a categorical variable, where each label would represent it's corresponding product.

Literature Review

General Domain - Literature Review

A less technical oriented recent publication by Jose Ramon Saura. (2021). [1] The author has demonstrated methods, frameworks & performance metrics in a less technical ways while leaning more on the value of data science. The published paper is intended to encourage more usage of data science driven analysis in digital marketing and to bridge the gap between the potential opportunities that can be achieved and the awareness of data science capabilities to support new digital marketing strategies within the community of digital marketing enthusiasts. In his second research question about the areas of further research and development in utilizing data science in digital marketing, he concluded 9 topics, of which one of them is Social Media Listening.

ML VS Domain Related - Literature Review

In a domain related research paper conducted by Microsoft Research Center Employees Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. [2] on the flip side of the coin, this research-based model was not constructed for the advertiser side but for the search engines to be a part of their advertising system where such prediction can help in enhancing their pricing models and provide insights and recommendations for the advertisers.
The authors proposed some modifications to the standard logistic regression algorithm to build a predictive model that can estimate the CTR of the new ads on search engine ads. The click through rates in digital marketing is a predictor of the ads appeal to users and it's computed by dividing the number of ad views over the clicks generated. Impressions/Clicks = CTR. CTR is an

almost equivalent to the Swipe Up Rate discussed in our research. Through out their research they tried several concepts, one of them is boosted regression trees where they say that it had no significance and they had to drop it. By reducing two primary source of variance, they were able to reduce the error rate with 30%.

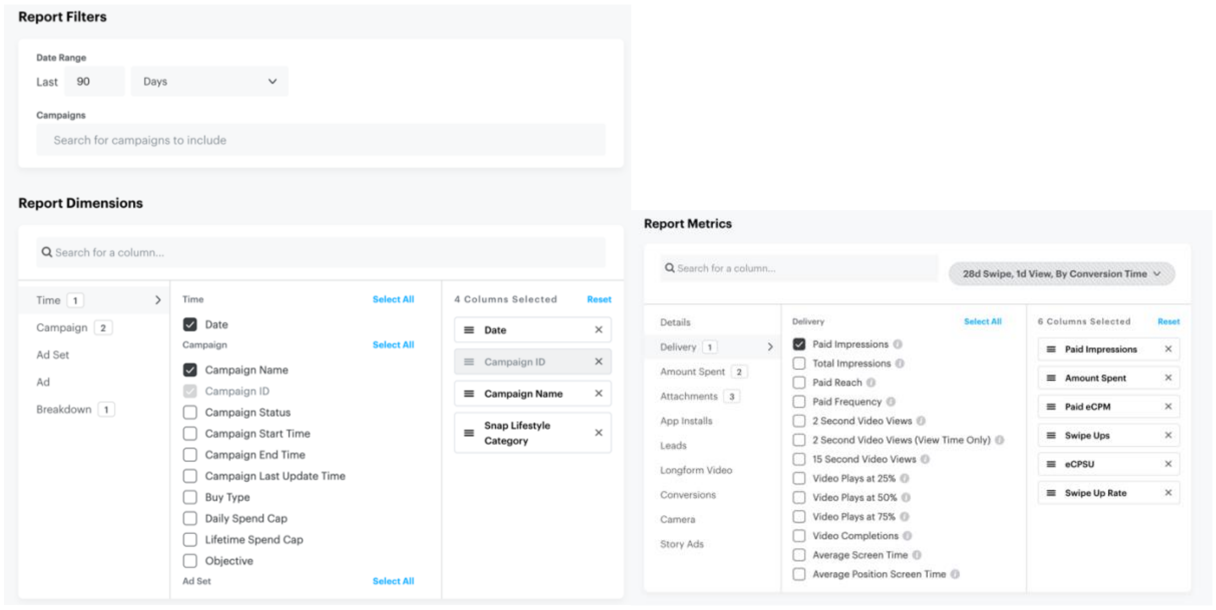
Parameters Tuning Methodologies - Literature Review

In one of the interesting papers published about tuning random forest parameters. Ramadhan, Muhammad & Sitanggang, Imas & NASUTION, Fahrendi & GHIFARI, Abdullah. (2017)[3] managed to increase the accuracy of their model from 0.9675 to 0. 96907 by applying grid search CV tuning on the Random forest algorithm based classification model that was intended to predict Gender based on Voice Frequency, the authors are claiming that the maximal accuracy was achieved by the need of tuning two parameters only which are the max_features & the n_estimators where they published a comparison table for six iterations of values passed to both parameters.

<u>Machine Learning Pipeline</u>

Data Extraction:

The data set was extracted manually from the snapchat ads manager dashboard using a private access, date filter was amended for 90 days time, Date, Campaign Name, Snap Lifestyle Category were selected from the dimension, Paid Impressions, Amount Spent, eCPM, Swip Ups, eCPSU & Swipe up Rate were initially selected from the Report Metrics. Later on some off these variables will be dropped due to the fact that they are just computed variables and would cause redundancy and bias to the accuracy, like for example swipe up rate is the percentage of swipe ups out of the impressions, CPSU or cost per swipe up is the Amount spent/Swipe Ups.


Report Filters

Date Range: Last 90 Days

Campaigns: Search for campaigns to include

Report Dimensions

Time 1	Campaign 2	Ad Set	Ad	Breakdown 1

Time: ☑ Date

Campaign: ☑ Campaign Name, ☑ Campaign ID, ☐ Campaign Status, ☐ Campaign Start Time, ☐ Campaign End Time, ☐ Campaign Last Update Time, ☐ Buy Type, ☐ Daily Spend Cap, ☐ Lifetime Spend Cap, ☐ Objective

Ad Set

4 Columns Selected: Date, Campaign ID, Campaign Name, Snap Lifestyle Category

Report Metrics

28d Swipe, 1d View, By Conversion Time

Details: Delivery 1, Amount Spent 2, Attachments 3, App Installs, Leads, Longform Video, Conversions, Camera, Story Ads

Delivery: ☑ Paid Impressions, ☐ Total Impressions, ☐ Paid Reach, ☐ Paid Frequency, ☐ 2 Second Video Views, ☐ 2 Second Video Views (View Time Only), ☐ 15 Second Video Views, ☐ Video Plays at 25%, ☐ Video Plays at 50%, ☐ Video Plays at 75%, ☐ Video Completions, ☐ Average Screen Time, ☐ Average Position Screen Time

6 Columns Selected: Paid Impressions, Amount Spent, Paid eCPM, Swipe Ups, eCPSU, Swipe Up Rate

Dataset Dictionary:

Independent Variables/Features	Definition
Start Time	A representation for the assoicated day for each observation.
Campaign Name	The advertising campaign name that belongs to the observation.
Lifestyle Category	The Lifestyle category of which this obesrvation is associated.
Paid Impressions	The amount of ad views. 1 Impression = 1 Ad View.
Paid eCPM	Estimated Cost Per Mile. 1 Mile = 1,000 Impressions = 1,000 Ad Views.
Amount Spent	The cost of each observation. Amount spent = (Paid Impressions/1000) eCPM
eCPSU	Estimated cost per Swipe UP. eCPSU = Amount Spent/Swipe Ups
Swipe Up Rate	The % of Swipe ups out of the Paid impressions

Dependant	Definition
Swipe Ups	Swiping up is the interaction that the user takes on Snapchat Platform to engage with some content. It can be looked at it the same we we look at clicks on the traditional platforms. Swipe Ups is going to be our target variable in all of the models.

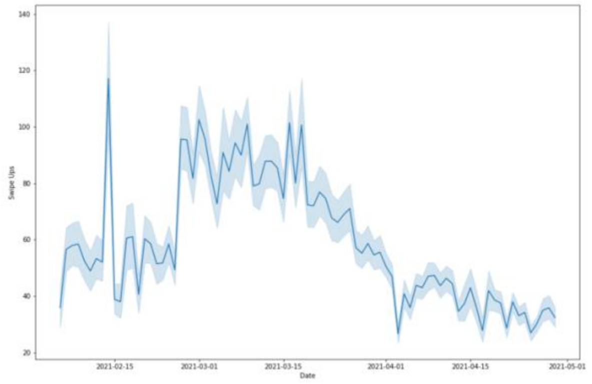
Data Exploration, Cleansing & Pre-Processing

From an initial visual review for the dataset on Excel, a few consecutive observations had zero values for all variables, this will be handled later in python.
The data set consists of 31,578 row observations & 9 columns.

1) Date Column

From the .head() function we noticed that the Start Time variable includes unnecessary time value of 00:00:00, a function was created to split date and time and stored in a new variable called date. Then a plot of Swipe Ups Vs. Date was generated to understand the Swipe up trend

```
In [8]: #### Exploratory Swipe ups Vs Date

In [9]: plt.figure(figsize=[15, 10])
        sns.lineplot(x=df['Date'], y=df['Swipe Ups'])
        plt.show()
```



2) Lifestyle Categories:

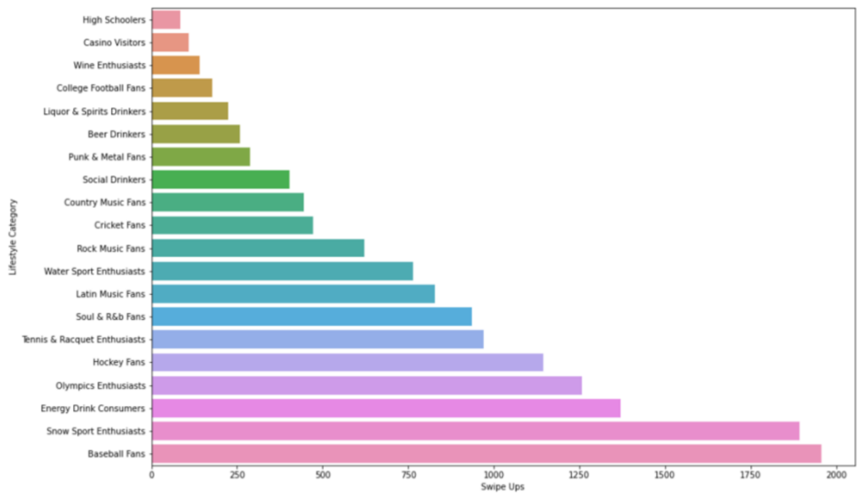
Matching with the dataset observations, there are 31,578 values for the lifestyle categories, however the unique values count is 118, which answers the question of how many categories are out there.

To get some insights on the top performing lifestyle categories and the lowest performing categories, a temp dataframe was created where a sum of each category swipe ups was calculated and then sorted the whole thing by values to to allow me to use indexing in plotting

```
In [27]: result = df.groupby(["Lifestyle Category"])['Swipe Ups'].aggregate(np.sum).reset_index().sort_values('Swipe Ups')
```

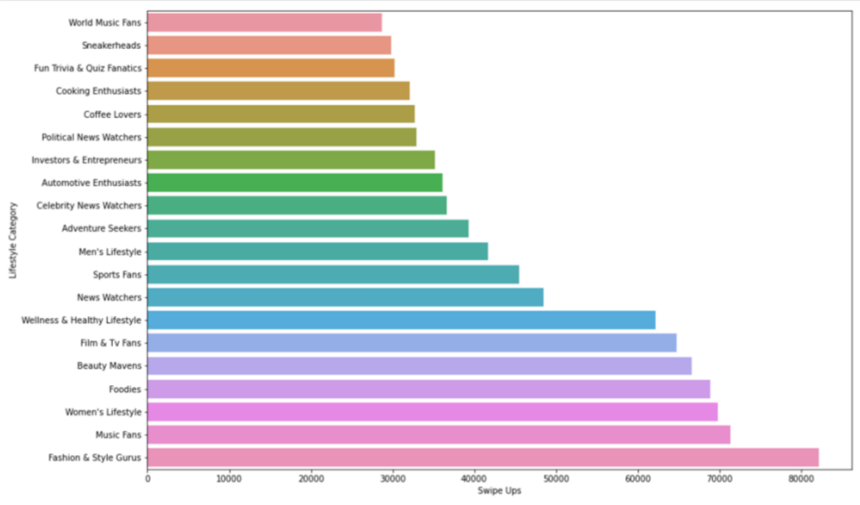
Top 20 Least Interactions

```
In [14]: plt.figure(figsize=[15, 10])
         sns.barplot(y=result['Lifestyle Category'][:20], x=result['Swipe Ups'][:20])
         plt.show()
```



Top 20 Most Interactions

```
In [16]: plt.figure(figsize=[15, 10])
         sns.barplot(y=result['Lifestyle Category'][-20:], x=result['Swipe Ups'][-20:])
         plt.show()
```



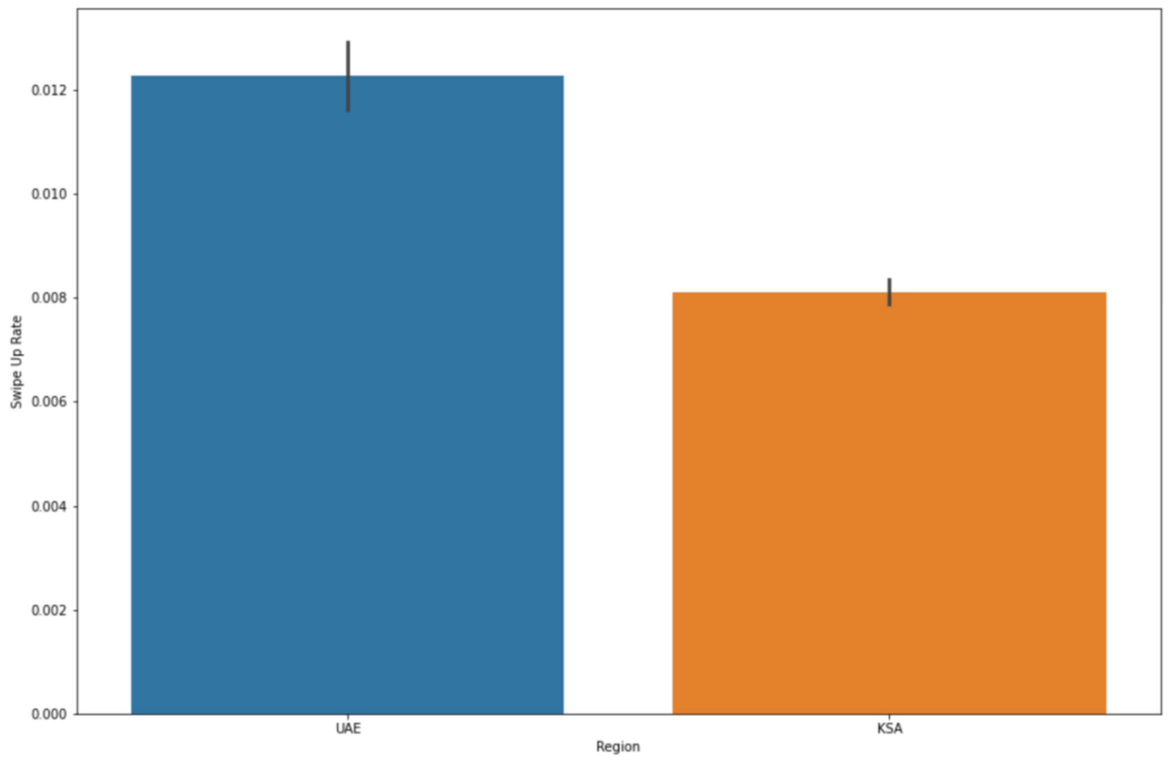
While the above plots are extremely insightful, we still can't fully depend on it for making category targeting decisions because originally the amounts of ad views served and the amount of budget spent to each category is not equal and was left for snap ads to make the choice based on its machine learning algorithm that works on predicting the better performing for each objective. However, in all of our campaigns, our objective was set to Swipe Ups.

3) Campaign Name

In the naming convention of the campaigns, if a campaign is targeting another country, the targeted country name should be mentioned in the campaign name, which allowed us to easily add a new categorical variable for the targeted region by creating an if function. As stated earlier, the research is covering only two countries in the GCC (KSA & UAE).

```
In [30]: ## Creating New column for country targeting of campaign
         def region(value):
             if "KSA" in value:
                 return "KSA"
             else:
                 return "UAE"

         df['Region'] = df['Campaign Name'].apply(region)
```

Another insightful plotting below, shows that campaigns within the UAE has higher swipe up rates than KSA, which makes sense because all of the advertised real estate products are in the UAE.



4) General Pre-Processing

- In the initial review of the dataset on excel, a few rows were observed to have zero values in the numerical variables , using the == 0 command, 224 rows were detected of which all of them were dropped.
- Label encoder was applied on the categorical variables (Campaign Name, Region & Lifestyle Category)
- Normalization on the numerical variables was applied using the MinMaxScaler
- If normalization resulted in null values, they were dropped as well.

5) Final Data Preparation for Modelling

Independent and dependent variables are separated in X_main & y_main, then were copied in X & y, the reason why we are doing this is to have a backup checkpoint that we can get back to without the need to rerun the code from the beginning.

Using function of train test split from the Sklearn library, the data were spitted into train and test with the size of 25% to the test.

<u>Modelling</u>

Several supervised regression models will be adopted in this research, initially Linear Regression, Decision Tree, Random Forest and SVR.

Due to the limited page count, rather than filling my document with definitions and theoretical explanation of each algorithm on its own, they will be explained and demonstrated through the results interpretation and parameters tuning.

Baseline

In order to have a benchmark for the research to measure how far we can achieve in the different modelling iterations, the DummyRegressor from sklearn library was used to provide basic predictions using simple rules. The chosen strategy is "mean" which denotes that the mean of the train set will be the score outcome.
The mean outcome will then be compared against the R2 scores and the mean scores of the employed models. As an indicator of potentiality, models should perform better than the baseline model.
The score of the baseline model is **-0.000149**, this result raises a lot of flags of which we are going to explore the reasons behind it in this research, rather than the extremely low score, the negative value is a major flag, which is discussed in the appendix.

Evaluation Parameters & Results

R-Squared:

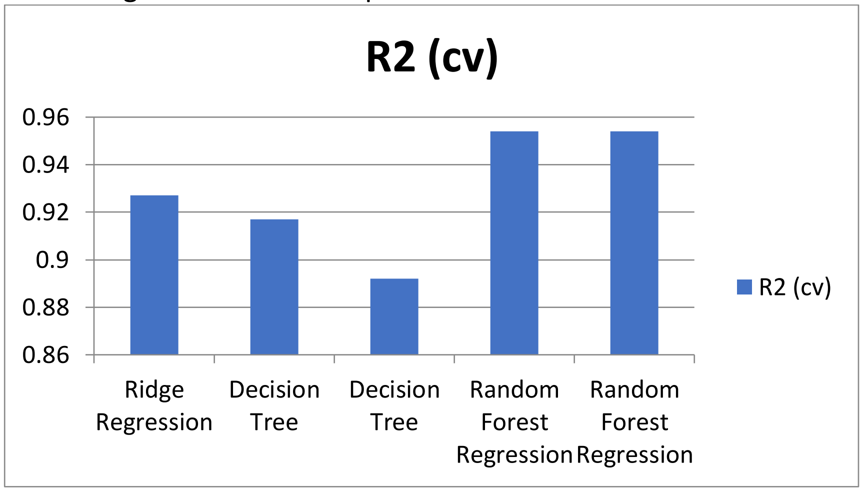
R^2 represents the variance of dependent variable explained by independent variables in a regression model. The formula of R^2 is:

$$\mathbf{R^2 = 1 - (Total\ Variation \backslash Unexplained\ Variation)}$$

R^2 was used as an important evaluation parameter as it indicates how well the model fits the data. High value of R^2 close to 100% indicates that the regression predictions perfectly fit within the data.

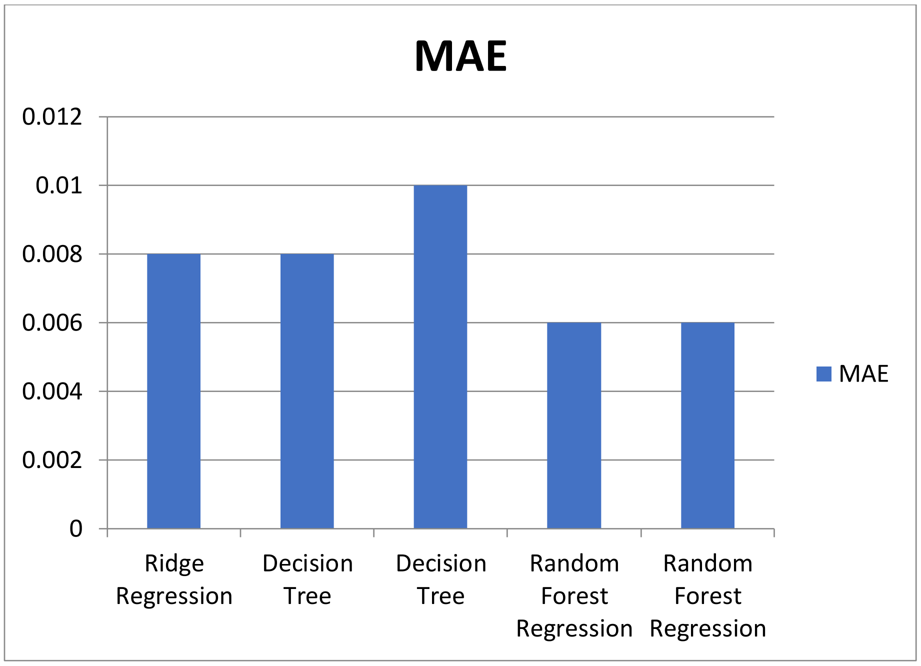
Cross Validation:

Cross-validation was used as well for splitting the data multiple times with different random state and provides back the average score for each split.



Random Forest captures the most variation and has an almost perfectly fitting model, followed by ridge regression and decision tree.

Mean Absolute Error:
Mean Absolute Error was another evaluation metric for the model, MAE measures the error between the predicted output and actual output. It is another crucial parameter to determine the performance of a regression model. The ideal value of MAE should be as low as possible, value close to zero means that the model is performing well. MAE is a insightful indicator to understand how accurately the model predicts.



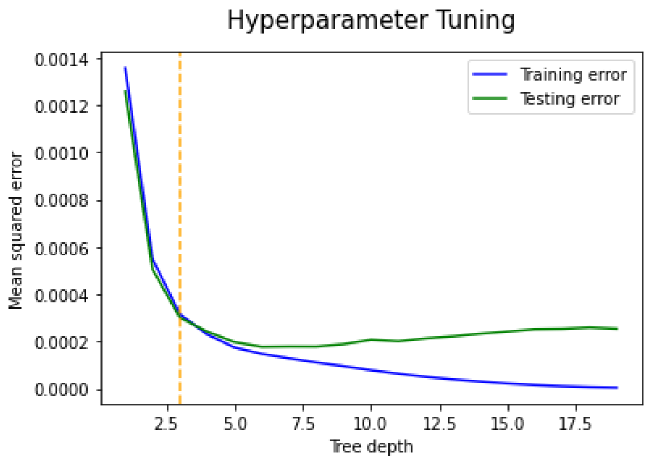
In terms of Mean Absolute Error, Random Forest Regressor is more accurate comparatively as it is closer to zero followed by Ridge Regression and Decision Tree.

Overfitting:
For overfitting or underfitting compliance check up, the score of both train and test was compared and below is a showcase for each model overfitting/underfitting scenario.
The Ridge linear regression did not report back any issues, with almost a perfect score, the difference between train and test score is 0.2% only.
Unlike linear regression, Decision Tree requires some tuning due to the wide over-fitting gap, the train score was 1 and the test was 0.9. An R^2 of 1 indicates that the regression predictions perfectly fit the training data but wasn't able to make equally accurate predictions on the test set. "An R-squared of 100% means that all movements of a dependent variable are completely explained by movements in the independent variable(s)" [4].
By applying the hyper parameter tuning method explained in this document for the decision tree, the accuracy score dropped to 0.89, however it resulted in a better difference between both the training and test sets which is 0.2%

Similarly, Random Forest with no tuning was overfitting, and post tuning resulted in better fitting, however the gap was still at 2%, which some might argue that it's normal and acceptable.

Hyper-parameter tuning

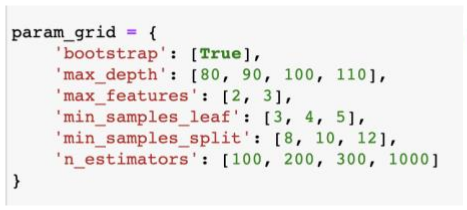
Decision Tree

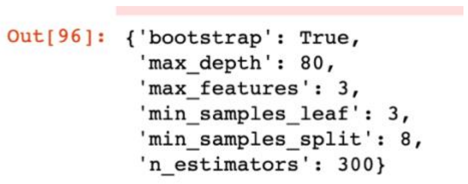


A key feature of decision tree is the depth; max_depth determines how far the nodes can expand. All values between 1 and 20 were test & plotted against the train and test error to determine the best value for tree depth. Clearly from the above plot, the tree depth = 3 was the best scenario where we had the least Mean squared error and the minimum gap between train and test error. Decision Tree I then trained my Decision tree with max_depth = 3 to check the new score. While this turning attempt resulted in lower accuracy score, it waived off the overfitting in a significant way.

Grid Search CV

After the Random forest showed promising results in terms of accuracy, it was decided to try and tune it with the GridSearchCV method to identify the best parameters that can be passed for the RandomForestRegressor where different combinations were tested, scores calculated and provided back the best parameters to be used.
As GridSearchCV requires a long computational time, *bootstrap* was adjusted for *true* value to use only a sample of data instead of the entire data set for faster results.
Different Values was used for the *max_depth* between *80 & 110 to* determine the optimal split of nodes
Max_features had two value trials 1 & 2 for the number of features a random forest is allowed to try.
N_estimators for the number of trees that are present in the random forest between 100 & 1000.
Minimum sample leaves for the number of samples required at the leaf node as well as minimum sample split for the number of data point present in each node before split.
The best combination received from GridSearchCV is as shown below.


```
param_grid = {
    'bootstrap': [True],
    'max_depth': [80, 90, 100, 110],
    'max_features': [2, 3],
    'min_samples_leaf': [3, 4, 5],
    'min_samples_split': [8, 10, 12],
    'n_estimators': [100, 200, 300, 1000]
}
```


```
Out[96]: {'bootstrap': True,
          'max_depth': 80,
          'max_features': 3,
          'min_samples_leaf': 3,
          'min_samples_split': 8,
          'n_estimators': 300}
```

Consolidate Results

Models	Tuning	Train R^2	Test R^2	R^2 (cv)	MAE
Ridge Regression		0.927	0.925	0.927	0.008
Support Vector Regression		-1.156	-1.317	-1.166	0.074
Decision Tree		1.0	0.90	0.917	0.008
Decision Tree	max_depth=3	0.894	0.892	0.892	0.010
Random Forest Regression		0.993	0.948	0.954	0.006
Random Forest Regression	max_depth= 80, max_features= 3, min_samples_leaf= 3, min_samples_split= 8, n_estimators= 300	0.979	0.950	0.954	0.006

Observations:

- Regression based machine learning models can react extremely differently on similar data set.
- Tuned Random Forest Regression has the most promising results when it comes to R2 and even though there is a slight over fitting, it has the lowest Mean Absolute Error, which means that it has the lowest error rate in prediction
- Linear Regression comes in the second place with quite good R2 scores as well and perfect fitting while the MAE is also very near to the best performing model.
- SVR showed a great failure, however parameters adjustment was not explored and SVR is very sensitive to parameters adjustment. [5]

Conclusion & Future Work

An excellent prediction model using tuned Random Forest Regression can be deployed as a product to predict the swipe ups for Snapchat campaigns taking into consideration the inconclusive dataset due to the limitations stated in the introduction. The variables of amount spent, and Lifestyle category interest can be manipulated to show their effect on the dependent variables where such tool can help in Digital Marketing by providing forecasting insights that would lead to better strategy planning.

For future work, another baseline models should be introduced to this research as the dummy regressor didn't show reliability.

Appendix

Negative R2 Explained

“This statistic measures how successful the fit is in explaining the variation of the data. Put another way, R-square is the square of the correlation between the response values and the predicted response values. It is also called the square of the multiple correlation coefficient and the coefficient of multiple determination.

Note that it is possible to get a negative R-square for equations that do not contain a constant term. Because R-square is defined as the proportion of variance explained by the fit, if the fit is actually worse than just fitting a horizontal line then R-square is negative. In this case, R-square cannot be interpreted as the square of a correlation. Such situations indicate that a constant term should be added to the model.” [5]

Bibliography

[1] Jose Ramon Saura, April 2021. Using Data Sciences in Digital Marketing: Framework, methods, and performance metrics. Journal of Innovation & Knowledge.
https://doi.org/10.1016/j.jik.2020.08.001.
(https://www.sciencedirect.com/science/article/pii/S2444569X20300329)

[2] Ramadhan, Muhammad & Sitanggang, Imas & NASUTION, Fahrendi & GHIFARI, Abdullah. (2017). Parameter Tuning in Random Forest Based on Grid Search Method for Gender Classification Based on Voice Frequency. DEStech Transactions on Computer Science and Engineering. 10.12783/dtcse/cece2017/14611.

[3] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In Proceedings of the 16th international conference on World Wide WebAssociation for Computing Machinery, New York, NY, USA, 521–530. DOI:https://doi.org/10.1145/1242572.1242643

[4] https://www.investopedia.com/terms/r/r-squared.asp

[5] https://web.maths.unsw.edu.au/~adelle/Garvan/Assays/GoodnessOfFit.html

Predicting how users with different lifestyle category interest will interact with off-plan multi-lingual Real Estate Video Ads in the GCC* on Snapchat¶

Importing Libraries¶

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing Dataset¶

In [2]:

```
df = pd.read_csv("set2.csv")
df.head()
```

Out[2]:

	Start Time	Lifestyle Category	Campaign Name	Paid Impressions	Amount Spent	Paid eCPM	Swipe Ups	eCPSU	Swipe Up Rate
0	2021-04-01 00:00:00 GST	Women's Lifestyle	Sobha - 3ayar - UAE	14158	55.67	3.93	56	0.99	0.0040
1	2021-04-01 00:00:00 GST	Automotive Shoppers	Sobha - 3ayar - UAE	16091	56.39	3.50	64	0.88	0.0040
2	2021-04-01 00:00:00 GST	Social Drinkers	Sobha - 3ayar - UAE	97	0.25	2.55	1	0.25	0.0103
3	2021-04-01 00:00:00 GST	Beer Drinkers	Sobha - 3ayar - UAE	39	0.15	3.89	0	0.00	0.0000
4	2021-04-01 00:00:00 GST	Wine Enthusiasts	Sobha - 3ayar - UAE	17	0.05	2.85	0	0.00	0.0000

In [3]:

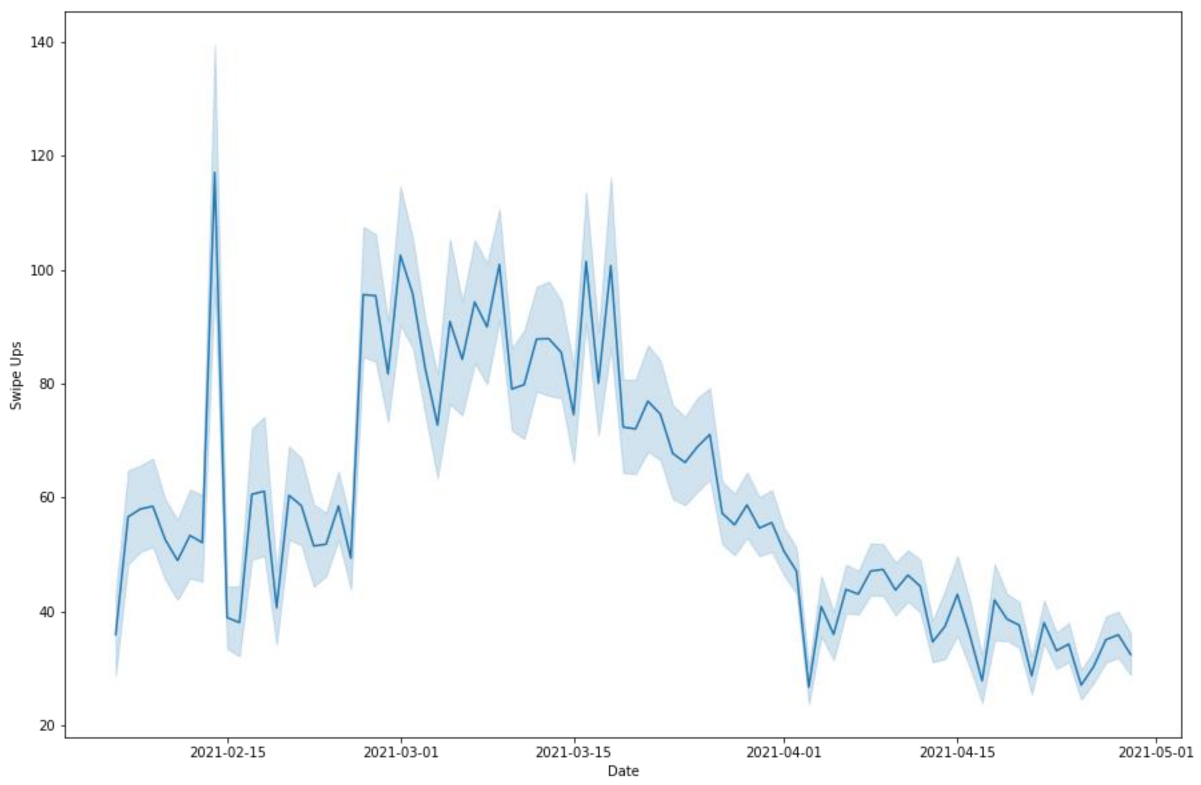
```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31578 entries, 0 to 31577
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
```

```
---  ------              --------------  -----
 0   Start Time          31578 non-null  object
 1   Lifestyle Category  31578 non-null  object
 2   Campaign Name       31578 non-null  object
 3   Paid Impressions    31578 non-null  int64
 4   Amount Spent        31578 non-null  float64
 5   Paid eCPM           31578 non-null  float64
 6   Swipe Ups           31578 non-null  int64
 7   eCPSU               31578 non-null  float64
 8   Swipe Up Rate       31578 non-null  float64
dtypes: float64(4), int64(2), object(3)
memory usage: 2.2+ MB
```
Dropping Values wherever Paid Impressions Amount Spent Paid eCPM are null¶

In [4]:
```
df.drop(df[df['Paid Impressions'].isnull().values].index, axis=0, inplace=True)
df.drop(df[df['Amount Spent'].isnull().values].index, axis=0, inplace=True)
df.drop(df[df['Paid eCPM'].isnull().values].index, axis=0, inplace=True)
```
Data Cleansing, Pre Processing And Exploratory¶

1) Date column¶

In [5]:
```
## Function for transforming Start Time column
def datetime_split(start_time):
    return start_time.split(" ")[0]

## Transforming Date Time Column
df['Date'] = pd.to_datetime(df['Start Time'].apply(datetime_split))
```
In [6]:
```
#### Exploratory Swipe ups Vs Date
```
In [7]:
```
plt.figure(figsize=[15, 10])
sns.lineplot(x=df['Date'], y=df['Swipe Ups'])
plt.show()
```



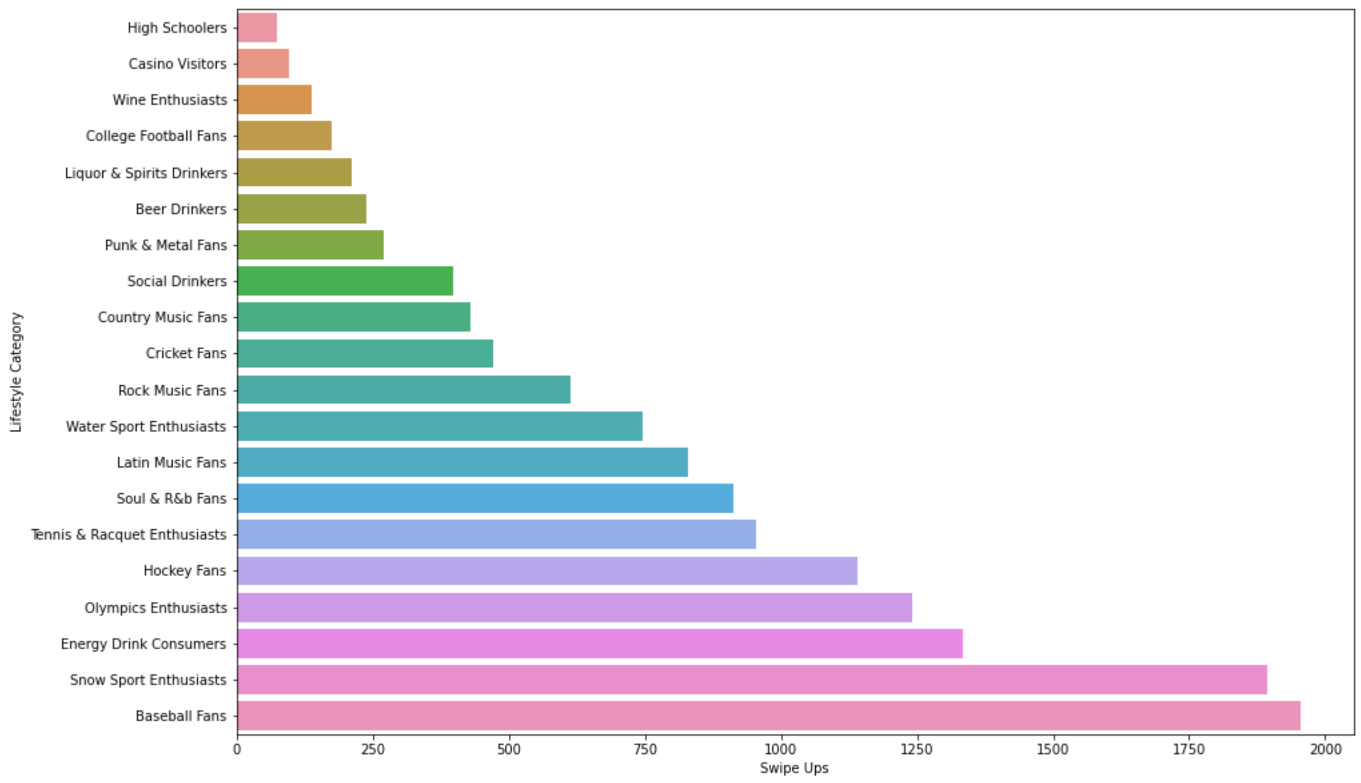
2) Lifestyle Category¶

In [8]:

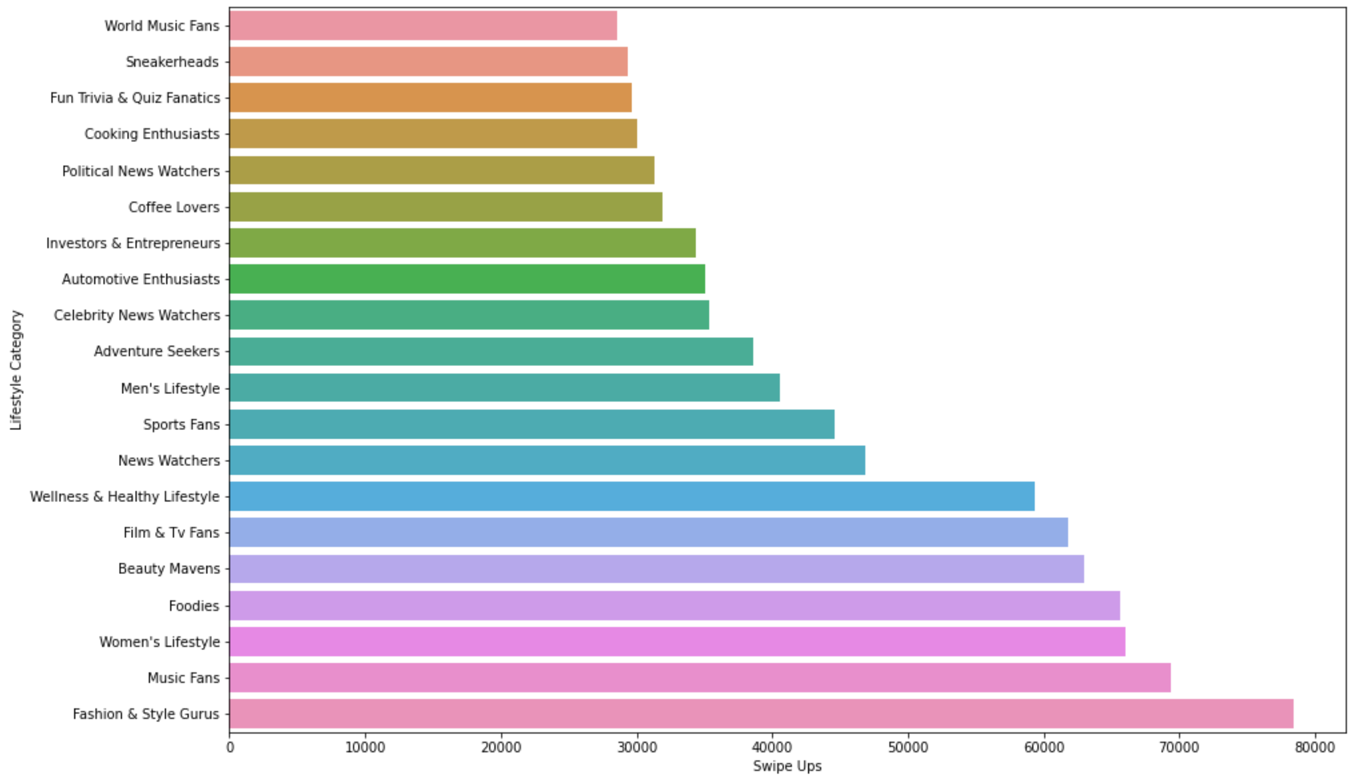
```
## Getting Unique values for Lifestyle column
df['Lifestyle Category'].unique()
```

Out[8]:

```
array(["Women's Lifestyle", 'Automotive Shoppers', 'Social Drinkers',
       'Beer Drinkers', 'Wine Enthusiasts', 'Liquor & Spirits Drinkers',
       'Fun Trivia & Quiz Fanatics', 'Meme Watchers',
       'Chat Fiction Enthusiasts', 'High Schoolers',
       'Advocates & Activists', 'Wellness & Healthy Lifestyle',
       'New Phone Seekers', 'Latin Music Fans', 'Esports Enthusiasts',
       'Soul & R&b Fans', 'World Music Fans', 'Cricket Fans',
       'Crime & Mystery Genre Fans', 'Indie & Foreign Film Fans',
       'Comics & Animation Fans', 'Talent & Competition Show Fans',
       'Do-it-yourselfers', 'Talk Show Fans', 'Fashion & Style Gurus',
       'Film & Tv Fans', 'Action & Thriller Genre Fans', 'Comedy Fans',
       'Cordcutters', 'Drama Genre Fans', 'Family Genre Fans',
       'Horror Genre Fans', 'Adventure Seekers', 'Arts & Culture Mavens',
       'Automotive Enthusiasts', 'Beachgoers & Surfers', 'Beauty Mavens',
       'Bookworms & Avid Readers', 'Clubbers & Party People',
       'Collegiates', 'Movie Theater Goers', 'Reality Tv Fans',
       'Romance & Rom-com Fans', 'Sci-fi & Fantasy Fans',
       'Superhero Film Fans', 'Fitness Enthusiasts',
       'Cycling Enthusiasts', 'Running Enthusiasts', 'Yoga Enthusiasts',
       'Foodies', 'Candy & Sweets Lovers', 'Cooking Enthusiasts',
       'Fast Food Junkies', 'Vegans & Organic Foodies', 'Gamers',
       'Casual & Mobile Gamers', 'Console & Pc Gamers',
       'Green Living Enthusiasts', 'Hipsters & Trendsetters',
       'Home Decoristas', 'Investors & Entrepreneurs',
       'Math & Science Enthusiasts', "Men's Lifestyle",
       'Sharp-dressed Men', 'Music Fans', 'Concert & Festival Goers',
       'Country Music Fans', 'Dance & Electronic Music Fans',
       'Indie & Alternative Music Fans', 'Pop Music Fans',
       'Punk & Metal Fans', 'Rock Music Fans', 'Hip-hop Music Fans',
       'News Watchers', 'Business News Watchers',
       'Celebrity News Watchers', 'Political News Watchers',
       'Outdoor & Nature Enthusiasts', 'Parents & Family-focused',
       'Pet & Animal Lovers', 'Philanthropists', 'Photographers',
       'Shoppers', 'Big Box Shoppers', 'Department Store Shoppers',
       'Luxury Shoppers', 'Online Shoppers', 'Toy Shoppers',
       'Sports Fans', 'College Basketball Fans', 'American Football Fans',
       'Baseball Fans', 'Basketball Fans', 'College Football Fans',
       'Fight & Wrestling Fans', 'Golfers', 'Hockey Fans',
       'Motor Sports Fans', 'Olympics Enthusiasts',
       'Snow Sport Enthusiasts', 'Teen & Young Adult Genre Fans',
       'Burger Lovers', 'Coffee Lovers', 'Pizza Lovers',
       'Shopping Mall Shoppers', 'Casino Visitors', 'Theme Park Visitors',
       'Soccer Enthusiasts', 'Sneakerheads', 'Street Sport Enthusiasts',
       'Energy Drink Consumers', 'Soft Drink Consumers',
       'Tennis & Racquet Enthusiasts', 'Water Sport Enthusiasts',
       'Techies & Gadget Fans', 'Travel Enthusiasts', 'Family Travelers',
       'Frequent Travelers'], dtype=object)
```
In [9]:
```
## Getting Unique values for Lifestyle column
df['Lifestyle Category'].count()
```
Out[9]:
```
31578
```
In [10]:
```
## Counting Unique values for Lifestyle column
df['Lifestyle Category'].nunique()
```
Out[10]:
```
118
```
In [11]:
```
### Preparing for Plotting
```
In [12]:
```
result = df.groupby(["Lifestyle Category"])['Swipe
Ups'].aggregate(np.sum).reset_index().sort_values('Swipe Ups')
```
In [13]:

Top 20 least interacted categories
In [14]:
```
plt.figure(figsize=[15, 10])
sns.barplot(y=result['Lifestyle Category'][:20], x=result['Swipe Ups'][:20])
plt.show()
```


In [15]:
Top 20
In [133]:
```
plt.figure(figsize=[15, 10])
sns.barplot(y=result['Lifestyle Category'][-20:], x=result['Swipe Ups'][-20:])
plt.show()
```



3) Campaign Name¶

In [17]:

```
## Getting Unique Values for Campaign Name
df['Campaign Name'].unique()
```

Out[17]:

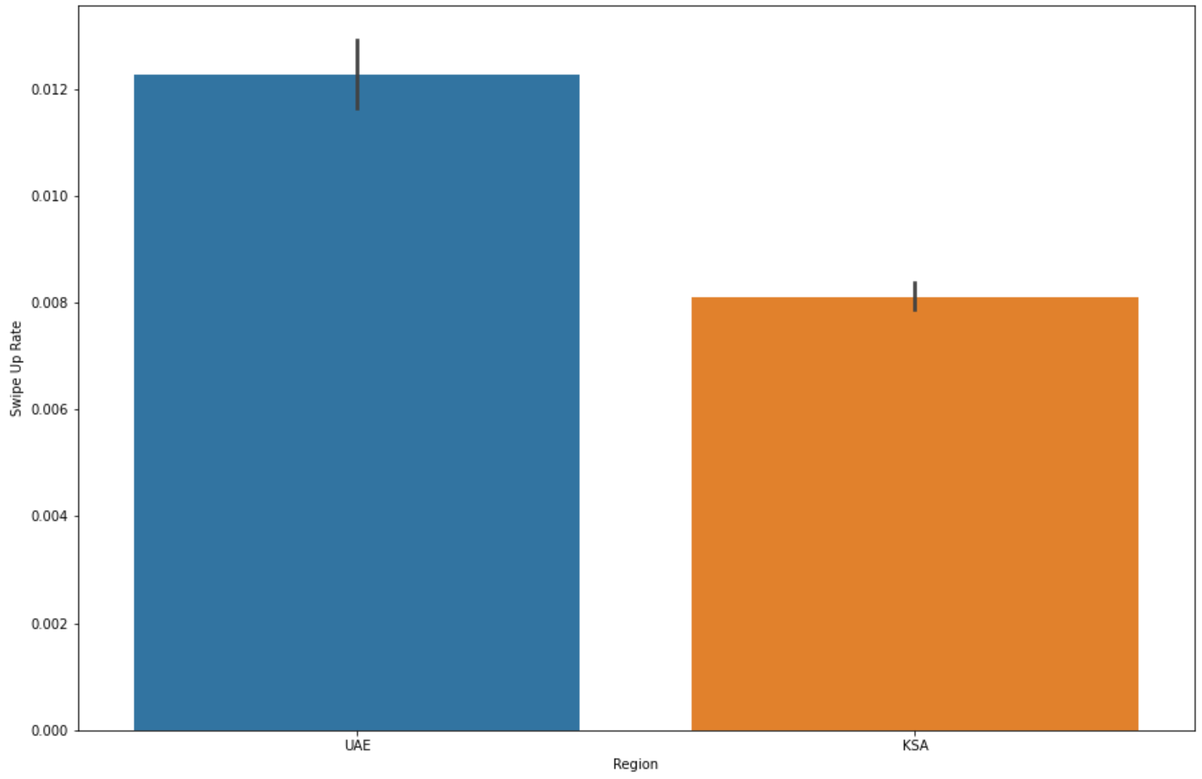
```
array(['Sobha - 3ayar - UAE', 'Binghatti - JVC', 'Aljurf - UAE - Native',
       'La Rosa - Dubai Prop', 'Pixel - UAE - Native',
       'Binghatti - Native', 'Binghati - KSA - Investors',
       'Sobha - UAE - Native', 'Aljurf - Jan - Native ', 'TG - Harmony 2',
       'Tilal alghaf - Ramadan', 'La Rosa - Targeted',
       'Falcon City Villas', 'Sobha - Static - UAE',
       'Waters Edge / Binghatti', 'Copy of Sobha - Jan - Native',
       'Binghatti JVC - Ramadan - UAE', 'Sobha - KSA - Investors'],
      dtype=object)
```

In [18]:

```
## Creating New column for country targeting of campaign
def region(value):
    if "KSA" in value:
        return "KSA"
    else:
        return "UAE"

df['Region'] = df['Campaign Name'].apply(region)
```

In [19]:

```
plt.figure(figsize=[15, 10])
sns.barplot(x=df['Region'], y=df['Swipe Up Rate'])
plt.show()
```



4) General Preprocessing¶

In [20]:

```
### Removing Rows that doesn't have values in numerical variables due to the system glitch
```

In [21]:

```
## Following rows contain no information in Numerical columns
## The rows that contain zero value for Paid Impressions, Amount Spent and Paid eCPM contain no value, therefore they will be dropped
df[(df['Paid Impressions'] + df['Amount Spent'] + df['Paid eCPM'] + df['Swipe Ups']) == 0]
```

Out[21]:

	Start Time	Lifestyle Category	Campaign Name	Paid Impressions	Amount Spent	Paid eCPM	Swipe Ups	eCPSU	Swipe Up Rate	Date	Region
9	2021-04-01 00:00:00 GST	High Schoolers	Sobha - 3ayar - UAE	0	0.0	0.0	0	0.0	0.0	2021-04-01	UAE
127	2021-04-02 00:00:00 GST	High Schoolers	Sobha - 3ayar - UAE	0	0.0	0.0	0	0.0	0.0	2021-04-02	UAE

	Start Time	Lifestyle Category	Campaign Name	Paid Impressions	Amount Spent	Paid eCPM	Swipe Ups	eCPSU	Swipe Up Rate	Date	Region
245	2021-04-03 00:00:00 GST	High Schoolers	Sobha - 3ayar - UAE	0	0.0	0.0	0	0.0	0.0	2021-04-03	UAE
341	2021-04-03 00:00:00 GST	Casino Visitors	Sobha - 3ayar - UAE	0	0.0	0.0	0	0.0	0.0	2021-04-03	UAE
356	2021-04-05 00:00:00 GST	Social Drinkers	Sobha - 3ayar - UAE	0	0.0	0.0	0	0.0	0.0	2021-04-05	UAE
...	...	...	...	...	...	...	...	...	...	...	...
27172	2021-04-26 00:00:00 GST	Casino Visitors	Binghatti JVC - Ramadan - UAE	0	0.0	0.0	0	0.0	0.0	2021-04-26	UAE
27194	2021-04-27 00:00:00 GST	High Schoolers	Binghatti JVC - Ramadan - UAE	0	0.0	0.0	0	0.0	0.0	2021-04-27	UAE
27290	2021-04-27 00:00:00 GST	Casino Visitors	Binghatti JVC - Ramadan - UAE	0	0.0	0.0	0	0.0	0.0	2021-04-27	UAE
27312	2021-04-28 00:00:00 GST	High Schoolers	Binghatti JVC - Ramadan - UAE	0	0.0	0.0	0	0.0	0.0	2021-04-28	UAE
27430	2021-04-29 00:00:00 GST	High Schoolers	Binghatti JVC - Ramadan - UAE	0	0.0	0.0	0	0.0	0.0	2021-04-29	UAE

224 rows × 11 columns

In [22]:

```
## Dropping Above rows
```

```
df.drop(df[(df['Paid Impressions'] + df['Amount Spent'] + df['Paid eCPM'] + df['Swipe Ups']) == 0].index, axis=0, inplace=True)
```
In [23]:
```
### Applying Label Encoder (Categorical Columns)


from sklearn.preprocessing import LabelEncoder

#### Label Encoder - Lifestyle Column

lb_lifestyle = LabelEncoder()
df['lifestyle_encoded'] = lb_lifestyle.fit_transform(df['Lifestyle Category'])

#### Label Encoder - Campaign Name

lb_campaign = LabelEncoder()
df['campaign_encoded'] = lb_campaign.fit_transform(df['Campaign Name'])

#### Label Encoder - Region

lb_region = LabelEncoder()
df['region_encoded'] = lb_region.fit_transform(df['Region'])

df.head()
```
Out[23]:

	Start Time	Lifestyle Category	Campaign Name	Paid Impressions	Amount Spent	Paid eCPM	Swipe Ups	eCPSU	Swipe Up Rate	Date	Region	lifestyle_encoded	campaign_encoded	region_encoded
0	2021-04-01 00:00:00 GST	Women's Lifestyle	Sobha - 3ayar - UAE	14158	55.67	3.93	56	0.99	0.0040	2021-04-01	UAE	115	11	1
1	2021-04-01 00:00:00 GST	Automotive Shoppers	Sobha - 3ayar - UAE	16091	56.39	3.50	64	0.88	0.0040	2021-04-01	UAE	6	11	1

	Start Time	Lifestyle Category	Campaign Name	Paid Impressions	Amount Spent	Paid eCPM	Swipe Ups	eCPSU	Swipe Up Rate	Date	Region	lifestyle_encoded	campaign_encoded	region_encoded
2	2021-04-01 00:00:00 GST	Social Drinkers	Sobha - 3ayar - UAE	97	0.25	2.55	1	0.25	0.0103	2021-04-01	UAE	97	11	1
3	2021-04-01 00:00:00 GST	Beer Drinkers	Sobha - 3ayar - UAE	39	0.15	3.89	0	0.00	0.0000	2021-04-01	UAE	11	11	1
4	2021-04-01 00:00:00 GST	Wine Enthusiasts	Sobha - 3ayar - UAE	17	0.05	2.85	0	0.00	0.0000	2021-04-01	UAE	114	11	1

In [24]:

```
#### Normalizing the Numerical columns

from sklearn.preprocessing import MinMaxScaler

## Extracting Numerical Columns
x = df[['Paid Impressions', 'Amount Spent', 'Paid eCPM', 'eCPSU', 'Swipe Up Rate', 'Swipe Ups'
]].values #returns an array with mumpy
min_max_scaler = MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
df_numerical = pd.DataFrame(x_scaled)
```

In [25]:

```
## Replacing the original values in dataset with normalized values from the above temp df
df['Paid Impressions'] = df_numerical[0]
df['Amount Spent'] = df_numerical[1]
df['Paid eCPM'] = df_numerical[2]
df['eCPSU'] = df_numerical[3]
df['Swipe Up Rate'] = df_numerical[4]
```

In [29]:

```
df['Swipe Ups'] = df_numerical[5]
```
In [30]:
```
# Dropping Null Values resulted from normalization
df.drop(df[df['Paid Impressions'].isnull().values].index, axis=0, inplace=True)
df.drop(df[df['Amount Spent'].isnull().values].index, axis=0, inplace=True)
df.drop(df[df['Paid eCPM'].isnull().values].index, axis=0, inplace=True)
df.drop(df[df['Swipe Ups'].isnull().values].index, axis=0, inplace=True)
df.drop(df[df['eCPSU'].isnull().values].index, axis=0, inplace=True)
df.drop(df[df['Swipe Up Rate'].isnull().values].index, axis=0, inplace=True)
```
In [31]:
```
df.head()
```
Out[31]:

	Start Time	Lifestyle Category	Campaign Name	Paid Impressions	Amount Spent	Paid eCPM	Swipe Ups	eCPSU	Swipe Up Rate	Date	Region	lifestyle_encoded	campaign_encoded	region_encoded	Swip Ups
0	2021-04-01 00:00:00 GST	Women's Lifestyle	Sobha - 3ayar - UAE	0.051223	0.066047	0.078131	0.035737	0.105544	0.0040	2021-04-01	UAE	115	11	1	0.035737
1	2021-04-01 00:00:00 GST	Automotive Shoppers	Sobha - 3ayar - UAE	0.058216	0.066902	0.069583	0.040842	0.093817	0.0040	2021-04-01	UAE	6	11	1	0.040842
2	2021-04-01 00:00:00 GST	Social Drinkers	Sobha - 3ayar - UAE	0.000351	0.000297	0.050696	0.000638	0.026652	0.0103	2021-04-01	UAE	97	11	1	0.000638

	Start Time	Lifestyle Category	Campaign Name	Paid Impressions	Amount Spent	Paid eCPM	Swipe Ups	eCPSU	Swipe Up Rate	Date	Region	lifestyle_encoded	campaign_encoded	region_encoded	Swip Ups
3	2021-04-01 00:00:00 GST	Beer Drinkers	Sobha - 3ayar - UAE	0.000141	0.000178	0.077336	0.000000	0.000000	0.000	2021-04-01	UAE	11	11	1	0.000000
4	2021-04-01 00:00:00 GST	Wine Enthusiasts	Sobha - 3ayar - UAE	0.000062	0.000059	0.056660	0.000000	0.000000	0.000	2021-04-01	UAE	114	11	1	0.000000

Preparing Data for Modelling¶

In [80]:

```
## Seperating independant varabiles (Features) an depandant variable  (Target Variable)
#X_main = df[['Paid Impressions', 'Amount Spent', 'lifestyle_encoded', 'campaign_encoded',
'region_encoded' ,'Paid eCPM', 'eCPSU', 'Swipe Up Rate' ]]
X_main = df[['Paid Impressions', 'Amount Spent' ,'Paid eCPM' , 'lifestyle_encoded',
'campaign_encoded', 'region_encoded' ]]
#X_main = df[['Paid Impressions', 'Amount Spent', 'lifestyle_encoded', 'campaign_encoded',
'region_encoded' ,'Paid eCPM' ]]


y_main = df[['Swipe Ups']]

X = X_main
y = y_main
```

In [81]:

```
from sklearn.model_selection import train_test_split

## Splitting the data into test and train with 25% as the test size
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=20)
print("Train Data Shape: ", X_train.shape, X_test.shape)
print("Test Data Shape: ", y_train.shape, y_test.shape)
```

```
Train Data Shape:  (21791, 6) (9339, 6)
Test Data Shape:  (21791, 1) (9339, 1)
In [82]:
# Checking the effect of this in compare to the shapes before hand.

print("Train Data Shape: ", X_train.shape, X_test.shape)
print("Test Data Shape: ", y_train.shape, y_test.shape)
Train Data Shape:  (21791, 6) (9339, 6)
Test Data Shape:  (21791, 1) (9339, 1)
```

Modelling¶

Baseline¶

```
In [83]:
from sklearn.dummy import DummyRegressor
dummy_regr = DummyRegressor(strategy="mean")
dummy_regr.fit(X_train, y_train)
dummy_regr.score(X_test, y_test)
Out[83]:
-0.00014907394519170225
```

Modeling¶

Linear Regression Via Stats Models¶

```
In [84]:
import statsmodels.api as sm
In [85]:
from sklearn.linear_model import LinearRegression
linear_regressor = LinearRegression(fit_intercept=True, normalize=False, copy_X=True)
In [86]:
model = sm.OLS(y_train, sm.add_constant(X_train)).fit()
print(model.summary())
                            OLS Regression Results
==============================================================================
Dep. Variable:              Swipe Ups   R-squared:                       0.929
Model:                            OLS   Adj. R-squared:                  0.928
Method:                 Least Squares   F-statistic:                 4.715e+04
Date:                Sun, 02 May 2021   Prob (F-statistic):               0.00
Time:                        02:20:36   Log-Likelihood:                 60809.
No. Observations:               21791   AIC:                        -1.216e+05
Df Residuals:                   21784   BIC:                        -1.215e+05
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
=======
                  coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const           0.0058      0.001     10.667      0.000       0.005       0.007
```

```
Paid Impressions      0.8890      0.015     59.904      0.000       0.860       0.918
Amount Spent          0.4223      0.015     28.369      0.000       0.393       0.451
Paid eCPM            -0.0133      0.004     -3.272      0.001      -0.021      -0.005
lifestyle_encoded -6.541e-06   2.95e-06     -2.218      0.027   -1.23e-05    -7.6e-07
campaign_encoded     -0.0007     1.9e-05    -37.286      0.000      -0.001      -0.001
region_encoded        0.0021      0.000      6.490      0.000       0.001       0.003
==============================================================================
Omnibus:                12494.001   Durbin-Watson:                   1.984
Prob(Omnibus):              0.000   Jarque-Bera (JB):          2205462.462
Skew:                       1.699   Prob(JB):                         0.00
Kurtosis:                  52.168   Cond. No.                     1.42e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.42e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Ridge Regression¶

In [87]:

```
from sklearn.linear_model import Ridge

clf_ridge = Ridge(alpha=1.0)

# training the model
clf_ridge.fit(X_train, y_train)

#results

print("Train score: ", clf_ridge.score(X_train, y_train))
print("Test score: ",clf_ridge.score(X_test, y_test))
```

```
Train score:  0.9279750856251843
Test score:  0.9253900427904115
```

In [88]:

```
from sklearn.linear_model import Ridge

clf_ridge = Ridge(alpha=1.0)

cv = RepeatedKFold(n_splits=5, n_repeats=3, random_state=1)
# evaluate model



scoring = ['neg_mean_absolute_error', 'r2']
for score in scoring:
    scores = cross_val_score(clf_ridge, X, y, scoring=score, cv=cv, n_jobs=-1)
    if score == "neg_mean_absolute_error":
        scores = absolute(scores)
        print('Mean MAE: %.3f (%.3f)' % (mean(scores), std(scores)))
    else:
        print('R2: %.3f (%.3f)' % (mean(scores), std(scores)))
Mean MAE: 0.008 (0.000)
R2: 0.927 (0.003)
```

Support Vector Regression¶

In [90]:

```
from sklearn.svm import SVR

clf_svr2 = SVR()

# training the model
clf_svr2.fit(X_train, y_train)

#results

print("Train score: ", clf_svr2.score(X_train, y_train))
print("Test score: ", clf_svr2.score(X_test, y_test))
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/sklearn/utils/validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
Train score:  -1.1568621481541523
Test score:  -1.317463641385062
```

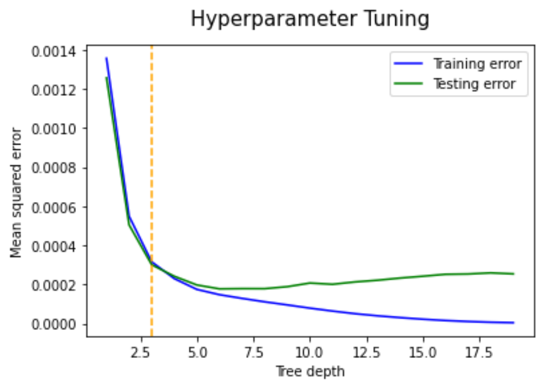
In [89]:

```
clf_svr = SVR()

cv = RepeatedKFold(n_splits=5, n_repeats=3, random_state=1)
# evaluate model
scoring = ['neg_mean_absolute_error', 'r2']
for score in scoring:
    scores = cross_val_score(clf_svr, X_train, y_train, scoring=score, cv=cv, n_jobs=-1)
    if score == "neg_mean_absolute_error":
        scores = absolute(scores)
        print('Mean MAE: %.3f (%.3f)' % (mean(scores), std(scores)))
    else:
        print('R2: %.3f (%.3f)' % (mean(scores), std(scores)))
Mean MAE: 0.074 (0.001)
R2: -1.166 (0.088)
```

Decision Tree¶

```
In [93]:
from sklearn.tree import DecisionTreeRegressor

dtr = DecisionTreeRegressor()

# training the model
dtr.fit(X_train, y_train)

#results

print("Train score: ", dtr.score(X_train, y_train))
print("Test score: ",dtr.score(X_test, y_test))
Train score:  1.0
Test score:  0.9049279777935209
In [134]:
# Decision Tree with cross Validation

dtr = DecisionTreeRegressor()

cv = RepeatedKFold(n_splits=5, n_repeats=3, random_state=1)
# evaluate model



scoring = ['neg_mean_absolute_error', 'r2']
for score in scoring:
   scores = cross_val_score(dtr, X, y, scoring=score, cv=cv, n_jobs=-1)
   if score == "neg_mean_absolute_error":
      scores = absolute(scores)
      print('Mean MAE: %.3f (%.3f)' % (mean(scores), std(scores)))
   else:
      print('R2: %.3f (%.3f)' % (mean(scores), std(scores)))
Mean MAE: 0.008 (0.000)
R2: 0.917 (0.006)
In [114]:
#Decision Tree Tuning
from sklearn.metrics import mean_squared_error

max_depths = range(1, 20)
training_error = []
for max_depth in max_depths:
   model_1 = DecisionTreeRegressor(max_depth=max_depth)
   model_1.fit(X, y)
   training_error.append(mean_squared_error(y, model_1.predict(X)))
```

```
testing_error = []
for max_depth in max_depths:
    model_2 = DecisionTreeRegressor(max_depth=max_depth)
    model_2.fit(X_train, y_train)
    testing_error.append(mean_squared_error(y_test, model_2.predict(X_test)))

plt.plot(max_depths, training_error, color='blue', label='Training error')
plt.plot(max_depths, testing_error, color='green', label='Testing error')
plt.xlabel('Tree depth')
plt.axvline(x=3, color='orange', linestyle='--')
plt.annotate('optimum = 3', xy=(3.2, 1.17), color='red')
plt.ylabel('Mean squared error')
plt.title('Hyperparameter Tuning', pad=15, size=15)
plt.legend()
plt.savefig('error.png')
```



In [113]:

```
# Rerun with tuning max depth

from sklearn.tree import DecisionTreeRegressor

dtr = DecisionTreeRegressor(max_depth=3)

# training the model
dtr.fit(X_train, y_train)

#results

print("Train score: ", dtr.score(X_train, y_train))
print("Test score: ",dtr.score(X_test, y_test))
```

```
Train score:  0.8948844046479922
Test score:  0.8921206042928919
```

In [135]:

```
# Decision Tree with cross Validation and Max Depth
```

```
dtr = DecisionTreeRegressor(max_depth=3)

cv = RepeatedKFold(n_splits=5, n_repeats=3, random_state=1)
# evaluate model



scoring = ['neg_mean_absolute_error', 'r2']
for score in scoring:
    scores = cross_val_score(dtr, X, y, scoring=score, cv=cv, n_jobs=-1)
    if score == "neg_mean_absolute_error":
        scores = absolute(scores)
        print('Mean MAE: %.3f (%.3f)' % (mean(scores), std(scores)))
    else:
        print('R2: %.3f (%.3f)' % (mean(scores), std(scores)))
Mean MAE: 0.010 (0.000)
R2: 0.892 (0.008)
```

Random Forest Regression¶

In [92]:

```
from sklearn.ensemble import RandomForestRegressor

clf_rf = RandomForestRegressor()

clf_rf.fit(X_train, y_train)
y_pred = clf_rf.predict(X_test)

print("Train score:   ", clf_rf.score(X_train,y_train))
print("Test score:   ", clf_rf.score(X_test,y_test))
<ipython-input-92-3e71746afdc1>:5: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  clf_rf.fit(X_train, y_train)
Train score:    0.9937744146417065
Test score:    0.9486830736922591
```

In [122]:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RepeatedKFold
from sklearn.model_selection import cross_val_score
from numpy import absolute
from numpy import mean
from numpy import std
```

```
clf_rf = RandomForestRegressor()

cv = RepeatedKFold(n_splits=5, n_repeats=10, random_state=1)


# evaluate model


scoring = ['neg_mean_absolute_error', 'r2']
for score in scoring:
    scores = cross_val_score(clf_rf, X_train, y_train, scoring=score, cv=cv, n_jobs=-1)
    if score == "neg_mean_absolute_error":
        scores = absolute(scores)
        print('Mean MAE: %.3f (%.3f)' % (mean(scores), std(scores)))
    else:
        print('R2: %.3f (%.3f)' % (mean(scores), std(scores)))
```

```
Mean MAE: 0.006 (0.000)
R2: 0.954 (0.004)
```

Optimizing with GridsearchCV¶

In [95]:

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'bootstrap': [True],
    'max_depth': [80, 90, 100, 110],
    'max_features': [2, 3],
    'min_samples_leaf': [3, 4, 5],
    'min_samples_split': [8, 10, 12],
    'n_estimators': [100, 200, 300, 1000]
}

# Create a based model
rf_gscv = RandomForestRegressor()
# Instantiate the grid search model
grid_search = GridSearchCV(estimator = rf_gscv, param_grid = param_grid,
                          cv = 3, n_jobs = -1, verbose = 2)
```

In [96]:

```
grid_search.fit(X_train, y_train)
grid_search.best_params_
```

```
Fitting 3 folds for each of 288 candidates, totalling 864 fits
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-
packages/sklearn/model_selection/_search.py:880: DataConversionWarning: A column-vector
y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for
example using ravel().
```

```
  self.best_estimator_.fit(X, y, **fit_params)
Out[96]:
{'bootstrap': True,
 'max_depth': 80,
 'max_features': 3,
 'min_samples_leaf': 3,
 'min_samples_split': 8,
 'n_estimators': 300}
In [100]:
#Re Running Random Forest using Grid Results

from sklearn.ensemble import RandomForestRegressor

clf_rf = RandomForestRegressor(
 max_depth= 80,
 max_features= 3,
 min_samples_leaf= 3,
 min_samples_split= 8,
 n_estimators= 300)

clf_rf.fit(X_train, y_train)
y_pred = clf_rf.predict(X_test)

print("Train score:   ", clf_rf.score(X_train,y_train))
print("Test score:   ", clf_rf.score(X_test,y_test))
<ipython-input-100-d0327e59a106>:12: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  clf_rf.fit(X_train, y_train)
Train score:    0.9790642036386032
Test score:    0.9505933916808247
In [123]:
# Rerunning Random forest using Grid Results Plus Cross Val Plus Negative mean scoring



clf_rf = RandomForestRegressor(
 max_depth= 80,
 max_features= 3,
 min_samples_leaf= 3,
 min_samples_split= 8,
 n_estimators= 300)

cv = RepeatedKFold(n_splits=5, n_repeats=10, random_state=1)
```

```
# evaluate model


scoring = ['neg_mean_absolute_error', 'r2']
for score in scoring:
  scores = cross_val_score(clf_rf, X_train, y_train, scoring=score, cv=cv, n_jobs=-1)
  if score == "neg_mean_absolute_error":
    scores = absolute(scores)
    print('Mean MAE: %.3f (%.3f)' % (mean(scores), std(scores)))
  else:
    print('R2: %.3f (%.3f)' % (mean(scores), std(scores)))
Mean MAE: 0.006 (0.000)
R2: 0.954 (0.004)
In [ ]:

In [ ]:

In [ ]:

In [126]:
!pip install --upgrade pip
Requirement already satisfied: pip in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages (21.0.1)
Collecting pip
 Downloading pip-21.1.1-py3-none-any.whl (1.5 MB)
   |████████████████████████████████| 1.5 MB 3.1 MB/s eta 0:00:01
Installing collected packages: pip
 Attempting uninstall: pip
  Found existing installation: pip 21.0.1
  Uninstalling pip-21.0.1:
   Successfully uninstalled pip-21.0.1
Successfully installed pip-21.1.1
In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```