Predicting Customer Next Visit

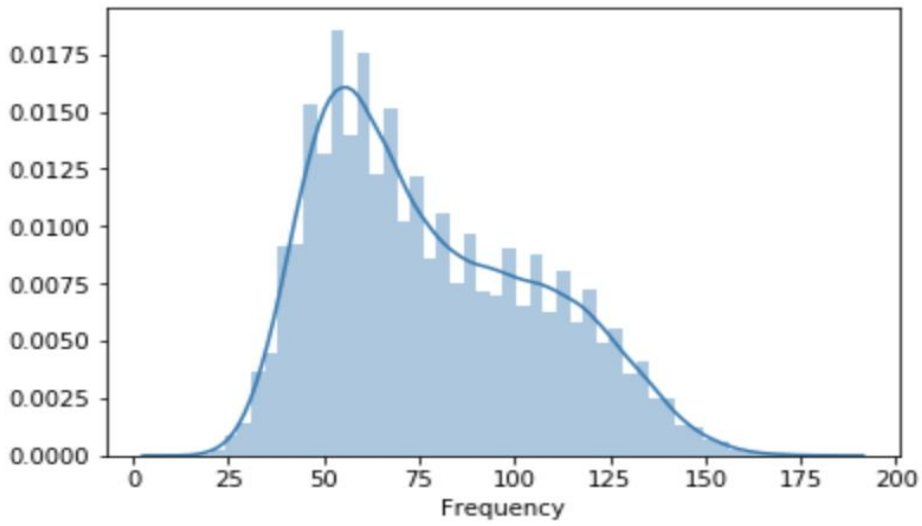
Business Problem

- Here we have a dataset with 300k rows.
- Each row represents a customer and with that we have a specific customer Information
- We are provided the days the customer has visited the shopping mall in the span of 1000 days
- As a business problem we have to predict on which weekday the customer will visit the mall in the week following the 1000th day.
- Solving this problem will help the organization to design the promotion packages for the target customers.

Assumption

As our dataset has visits in terms of day number ranging from 1 to 1001, we have made this assumption that 1 is the beginning of the week i.e. Monday. Thus, we have assigned numbers to the weekday as:
1: Monday
2: Tuesday…..
7: Sunday

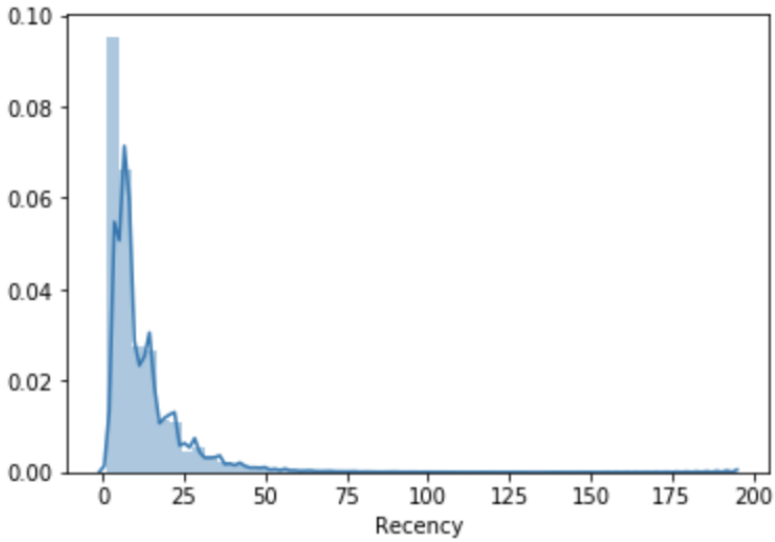
EDA - Frequency of visit



As seen in this graph, the number of times customer has visited the mall in the span of 1000 days lies in the range of 25-150 days.

There are some customers who have visited the mall more than 150 days. However, maximum of the customers is visiting 25-75 days in that span.

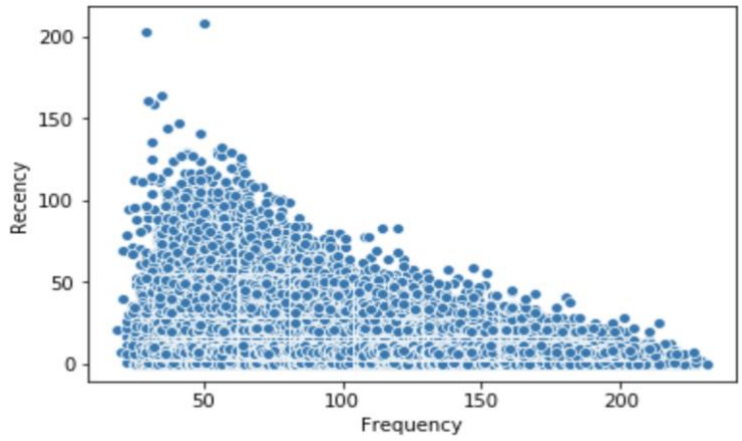
EDA – Recency



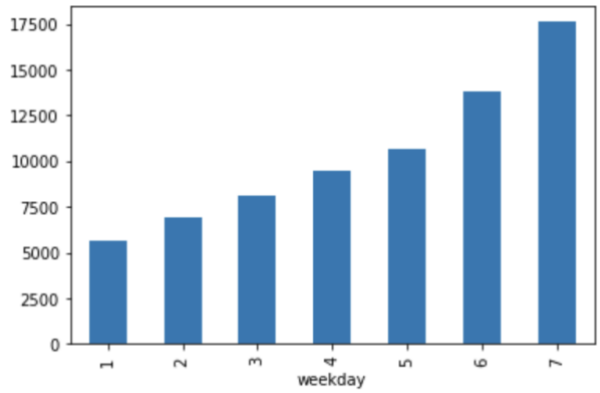
Recency is the how many days back the customer has visited the mall.
As per the graph, maximum customers have recently visited the mall.

Relationship between the recency and frequency

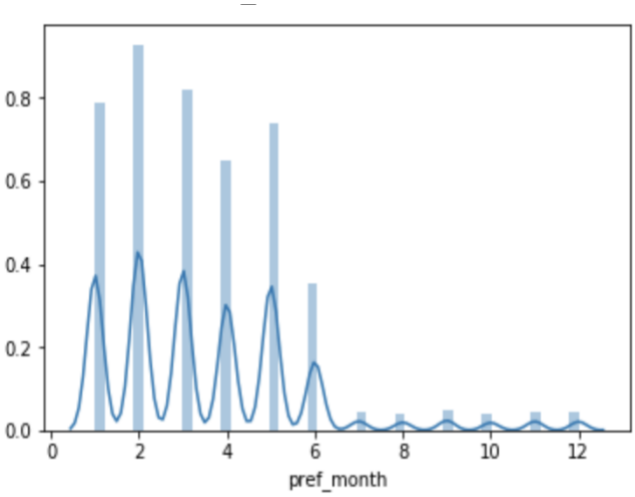
As seen in this plot, the customers who have high frequency have visited recently to the mall.
Thus the customers on the right side of the plot are more loyal.

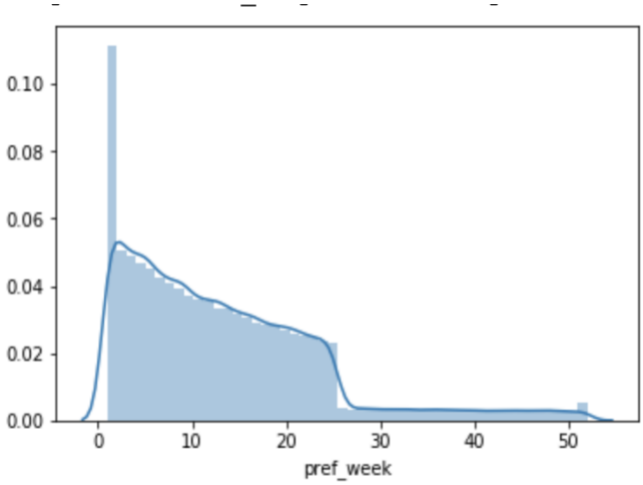


EDA - Most visited weekday of the week



Preferred month and Year of visit





Feature Engineering

Since in this dataset, we do not have many features given to us, so I have created some features of my own using the data I have.

Some basic features created are:
1. Mean difference between the days of visit.
2. Standard deviation of the difference between the days of visit.
3. Frequency of visit.

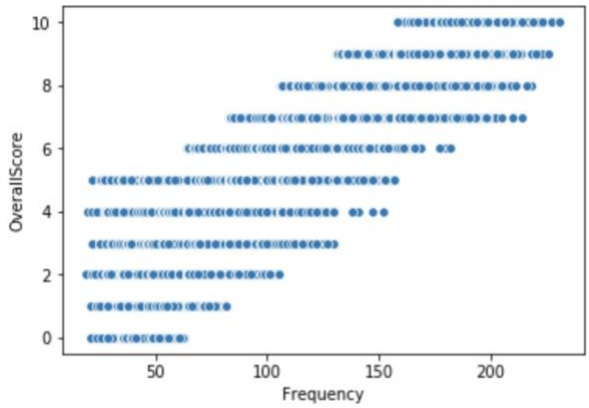
Feature Engineering -- K means

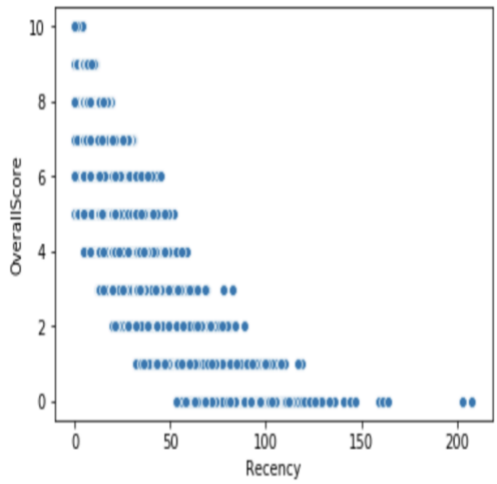
FrequencyCluster	count	mean	std	min	25%	50%	75%	max
0	46368.0	54.363160	7.047191	19.0	50.0	56.0	60.0	63.0
1	72379.0	72.880380	5.389658	64.0	68.0	73.0	77.0	82.0
2	61059.0	93.043941	6.591834	83.0	87.0	93.0	99.0	105.0
3	47580.0	117.682598	7.252034	106.0	111.0	118.0	124.0	130.0
4	45262.0	143.482988	7.734475	131.0	137.0	143.0	150.0	157.0
5	27352.0	171.599664	11.184586	158.0	163.0	169.0	178.0	231.0

RecencyCluster	count	mean	std	min	25%	50%	75%	max
0	2689.0	66.674600	14.679694	53.0	56.0	62.0	72.0	193.0
1	12277.0	38.479677	5.755671	31.0	34.0	37.0	42.0	52.0
2	29744.0	23.415714	3.346445	19.0	21.0	22.0	26.0	30.0
3	51291.0	13.996998	2.019188	11.0	12.0	14.0	15.0	18.0
4	94168.0	7.208330	1.446711	5.0	6.0	7.0	8.0	10.0
5	109831.0	2.017682	1.106525	1.0	1.0	2.0	3.0	4.0

	Recency								Frequency							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	ma
OverallScore																
0	1500.0	69.110667	17.099006	53.0	57.0	63.0	75.0	208.0	1500.0	51.000667	8.336446	21.0	46.0	52.0	58.0	6
1	5063.0	43.369741	10.996658	32.0	35.0	41.0	48.0	119.0	5063.0	55.654355	9.895913	21.0	49.0	56.0	61.0	8
2	10763.0	30.351203	9.196640	20.0	23.0	28.0	35.0	89.0	10763.0	60.856917	11.587838	19.0	53.0	60.0	69.0	10
3	18365.0	21.204084	7.600546	12.0	15.0	20.0	25.0	83.0	18365.0	65.651674	13.206899	22.0	57.0	65.0	75.0	13
4	30120.0	13.832802	6.939322	5.0	8.0	13.0	18.0	59.0	30120.0	69.388977	15.826650	20.0	58.0	68.0	79.0	15
5	44648.0	8.080855	6.194881	0.0	4.0	7.0	11.0	52.0	44648.0	73.845346	17.682749	22.0	62.0	72.0	82.0	15
6	50110.0	5.448932	5.354813	0.0	1.0	4.0	8.0	45.0	50110.0	86.198423	17.693350	64.0	73.0	82.0	96.0	18
7	46027.0	4.299520	4.710580	0.0	1.0	2.0	7.0	31.0	46027.0	105.242792	18.039272	83.0	91.0	101.0	117.0	21
8	39723.0	3.516149	3.760986	0.0	0.0	2.0	6.0	19.0	39723.0	127.439796	16.052265	106.0	115.0	124.0	138.0	21
9	34407.0	2.288633	2.711573	0.0	0.0	1.0	3.0	11.0	34407.0	148.683582	13.297824	131.0	139.0	147.0	155.0	22
10	19274.0	1.013593	1.189955	0.0	0.0	1.0	2.0	4.0	19274.0	172.080056	11.393873	158.0	163.0	170.0	179.0	23

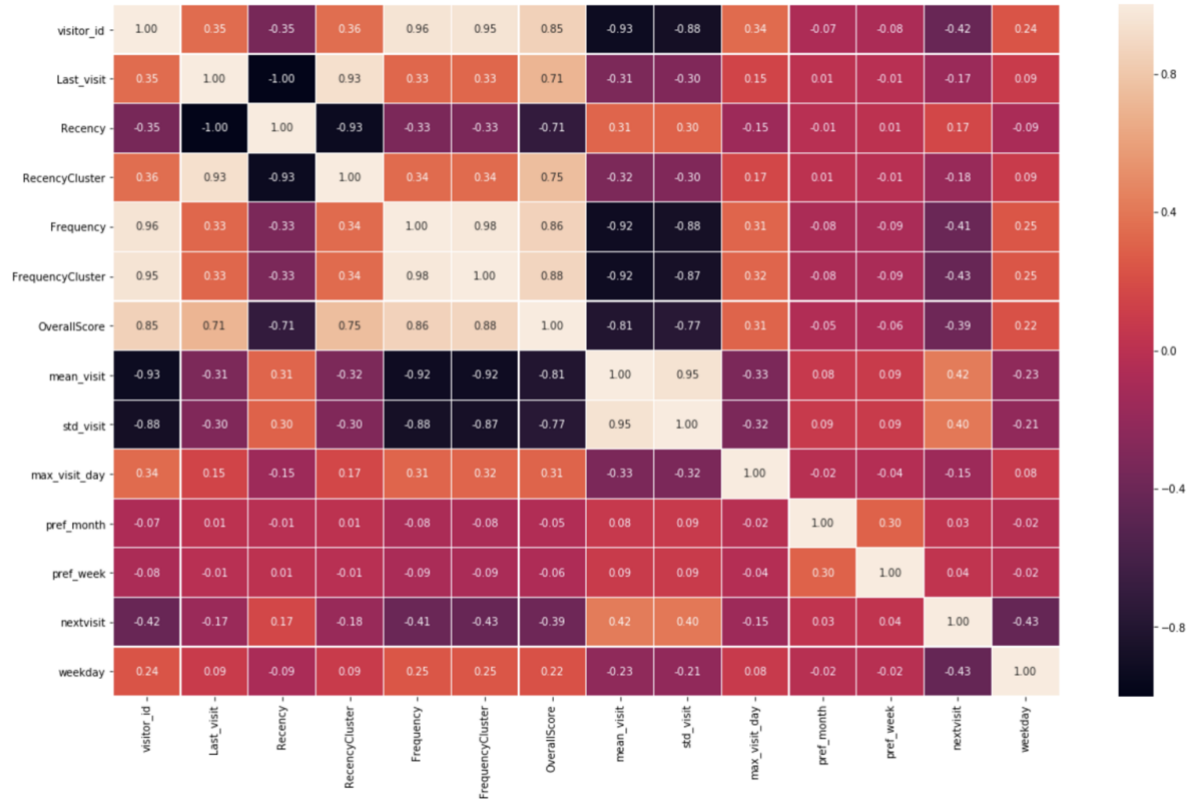
CLV with respect to recency and frequency





Feature elimination

After studying correlation between the feature in the dataset, we have dropped certain features which are highly correlated and make the learning model bias.



Target variable : Day of the week
Value ranges from 0-7
0 : user has not visited the mall in 130th week
1-7 : day of the week [monday : sunday]

Model 1 : Basic Xg Boost.

```
array([[45527,     0,     0,     1,     0,     0,     0,     0],
       [ 1142,     0,     0,     0,     0,     0,     1,     0],
       [ 1369,     0,     0,     0,     0,     0,     0,     0],
       [ 1673,     0,     0,     0,     0,     0,     0,     0],
       [ 1977,     0,     0,     0,     0,     0,     0,     0],
       [ 2094,     0,     0,     0,     0,     0,     0,     0],
       [ 2790,     0,     0,     0,     0,     0,     0,     0],
       [ 3426,     0,     0,     0,     0,     0,     0,     0]])
```

Accuracy : 76%..

Well, if we look into the target variable in y_train more than 70% of the data has value 0. I.e. the customer has not visited the mall that week.
Dataset is highly imbalanced and thus the bias comes and model learnt only to predict 0's.

Model 2: Balanced Xg Boost Model

We assigned weights to the models so that maximum occuring variable has minimum weight.

Accuracy : 48%

Confusion matrix:

```
array([[27180,   107,  6919,  1998,   624,   322,  2102,  6276],
       [  273,    16,   442,   100,    53,    18,    80,   161],
       [  360,    13,   493,   121,    51,    21,   107,   203],
       [  504,     7,   613,   135,    62,    23,    93,   236],
       [  572,    11,   732,   145,    85,    35,   109,   288],
       [  655,    11,   724,   160,    73,    31,   129,   311],
       [  868,    21,   878,   213,    96,    38,   186,   490],
       [ 1186,    15,  1036,   234,    97,    24,   231,   603]])
```

Hyper Parameter Tuning

We opted Grid Search for hyper parameter Tuning to improve the parameters value in XgBoost and get the best accuracy.
After running hyper parameter tuning we found the best parameters as :
({'max_depth': 3, 'min_child_weight': 1}, 0.46038323026677314)

Predicting Customer Visit in 144th week

```
Counter({0: 174556,
         1: 980,
         2: 40587,
         3: 21392,
         4: 23766,
         5: 1447,
         6: 6043,
         7: 31229})
```

We are able to predict this:
This shows that 980 customers are visiting on weekday 1