Executive Summary :

Problem Statement : Imagine being hungry in an unfamiliar part of town and getting restaurant recommendations served up, based on your personal preferences, at just the right moment. The recommendation comes with an attached discount from your credit card provider for a local place around the corner!

Right now, Elo, one of the largest payment brands in Brazil, has built partnerships with merchants in order to offer promotions or discounts to cardholders. But do these promotions work for either the consumer or the merchant? Do customers enjoy their experience? Do merchants see repeat business? Personalization is key.

Elo has built machine learning models to understand the most important aspects and preferences in individual or profile. We need to do similar.

Understanding the business problem :

- There are many factors that are tied to a customer's experience while they buy some products. While these factors are dependent on customer choices , they impact the businesses which is why studying customer's behaviours is so important for organizations like ELO so as to achieve Customer satisfaction while also expanding their businesses by identifying the right customers for right merchants and vice versa.

Motivation :

- We want to predict in customer loyalty.
- Our model will provide personalized experience to Elo's Customers , and help Elo reduce unwanted campaigns, while creating the right experience for customers.

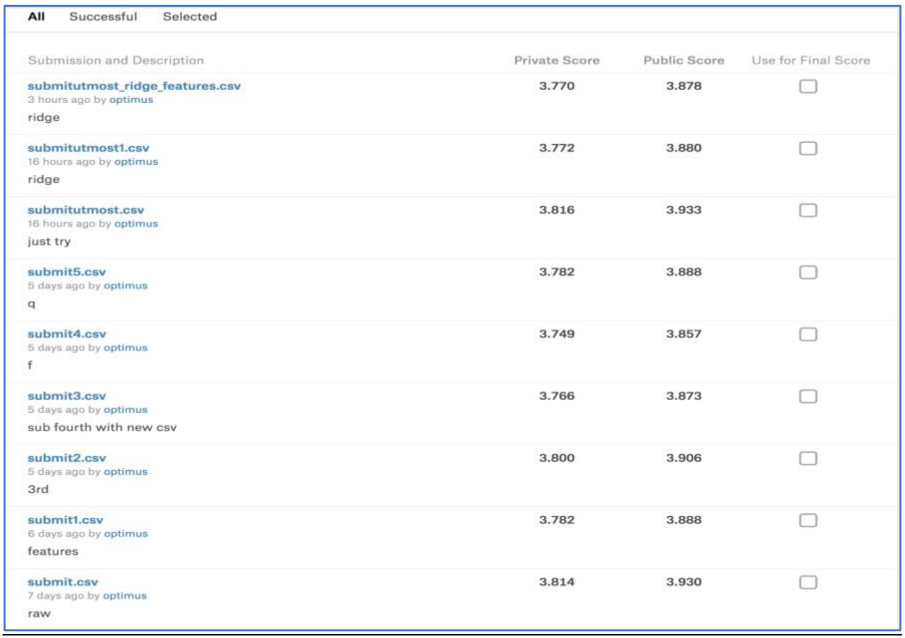
Approach :

- **RFM** (Recency, Frequency, Monetary) analysis is a proven marketing model for behaviour-based customer segmentation. It groups customers based on their transaction history – how recently, how often and how much did they buy. We tried to incorporate this while doing our feature extraction \ engineering.
- Looking at the dataset we clearly see that this is a big data problem , moreover since there are so many features involved , it could have been possible to come up with a straight forward programming model that would work for this problem. It's a prediction based problem hence we understand it to be in the realm of data- science
- We tried to do the following while looking at the issue at hand :
    - **EDA**:Did EDA on transactions and merchants files which involved understanding the business problem , studying the data , looking at the missing values and as required imputing them. Cleaning up duplicates and understanding the distribution profile as well as studying the correlation amongst different fields and what they mean was also a part of this exercise. We used the data dictionary and other details provided on Kaggle to understand the dataset better
    - **Feature Engineering and Transformations** : Once we addressed and cleaned the dataset , we tried to extract features while also merging the different datasets with the relevant keys as described in the ipynb files. Feature expansion and reduction both came into this realm. We tried to use our RFM and limited business knowledge to come up with new features. A classic example would be to derive the number of transactions a customer did.
    - **Transformations** were done as a two-step process , while we aggregated the data and came up with new features, considering it to be the primary transformation , we also did transformations like log , sqrt , box-cox,PCA while modelling.
    - **Model Selection** and optimization : We kept our focus on trying different variants of Linear Regression (making our assumption of this being a Linear Regression problem, considering assignment ask, PS : It clearly is not the case now that we have done it.). We used various techniques like Linear Regression , Lasso, Ridge , OLS , Ridge with a flavour of model optimization based on learning rate. We went back and forth with different features based on the VIF , and profiling outputs.We used K-Fold with K= 5 as our validation technique.
    - **Optimization** : We ran all the algorithms on the entire aggregated dataset, with only exception of OLS where we had to take the aggregated sample due to memory limitations of our machines.(is the reason why we see a bit higher RMSE for the same)
    - **In-Sample v/s Out of Sample tests :** While we did K-fold and generated RMSE for the models on the training set , we also used the test set provided by Kaggle to predict the scores and upload on Kaggle to get the out-of sample RMSE and check how our various models performed on the unseen data. Both the model RMSEs are summarized in below tables , one from the ipynbs and other from Kaggle submissions
    - **Diagnostics :** We finally use diagnostic plots to determine if our linear model adheres to the assumptions that a linear model should adher to :
        - Linear specification is correct: $Y = \beta_0 + \beta_1 X + \varepsilon$
        - Errors are uncorrelated with explanatory variables: $E[\varepsilon|X] = 0$
        - Constant variance of errors: $Var[\varepsilon_i]= \sigma_\varepsilon^2$
        - Errors are independent of each other: $Corr[\varepsilon_i, \varepsilon_j] = 0$
        - Error term is normally distributed: $\varepsilon_i \sim N(0, \sigma_\varepsilon^2)$
        - In addition, no predictor variable should be a linear combination of other predictor variables – no multicollinearity

Model Scores :

Model Used	Best RMSE from K-Fold where K=5 (In-Sample RMSE)	R-Squared	Comments
Linear Regression (SK Learn)	3.73251	0.0250	Simple linear model with all the features that we generated after aggregation and feature engineering
Linear Regression (SK Learn) after VIF and removing columns as per VIF	3.73258	0.0250	Linear model , with features with high vif considered for removal
Linear Regression (SK Learn) after PCA	3.780	-0.0001	PCA with 0.99 explainability and two components
Lasso (SK Learn)	3.7358	0.0233	Lasso : L1 regularization(absolute sum of co-efficients
Ridge (SK Learn)	3.73255	0.0250	Ridge L2 regularization ()sum of squares of co-efficients)
Linear Regression (SK Learn) with target variable adjusted on X	1.6494	0.8196	Overfitting by introducing a feature that tried to explain Y . We tried uploading the predicted values on test dataset on Kaggle and found that this lead to higher RMSE which leads to our conclusion that we should not try to overfit / explain Y by imputing its distribution
RIDGE (SK LEARN) : *TWEAKED*	3.7286	0.0270	Ridge with Feature selection as per profiling report , transformation and model optimization on learning rate
OLS(StatsModels)	3.84	0.02	OLS statistical linear Model

Kaggle Scores :



All Successful Selected

Submission and Description	Private Score	Public Score	Use for Final Score
submitutmost_ridge_features.csv 3 hours ago by optimus ridge	3.770	3.878	☐
submitutmost1.csv 16 hours ago by optimus ridge	3.772	3.880	☐
submitutmost.csv 16 hours ago by optimus just try	3.816	3.933	☐
submit5.csv 5 days ago by optimus q	3.782	3.888	☐
submit4.csv 5 days ago by optimus f	3.749	3.857	☐
submit3.csv 5 days ago by optimus sub fourth with new csv	3.766	3.873	☐
submit2.csv 5 days ago by optimus 3rd	3.800	3.906	☐
submit1.csv 6 days ago by optimus features	3.782	3.888	☐
submit.csv 7 days ago by optimus raw	3.814	3.930	☐

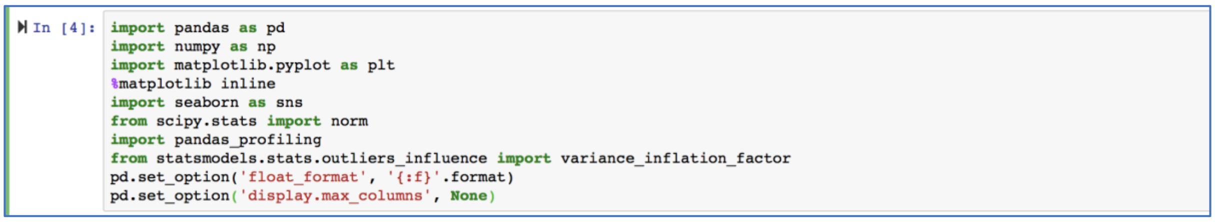
Methodology & Tools :

- RFM model / RFM Analysis for Customer Segmentation
  - RFM (Recency, Frequency, Monetary) analysis is a proven marketing model for behaviour-based customer segmentation. It groups customers based on their transaction history – how recently, how often and how much did they buy.
  - Recency (R) – How many days ago customer made a purchase? Deduct most recent purchase date from today to calculate the recency value. 1 day ago? 14 days ago? 500 days ago?
  - Frequency (F) – How many times has the customer purchased from our store? For example, if someone placed 10 orders over a period of time, their frequency is 10.
  - Monetary (M) – How many $$ has this customer spent? Simply total up the money from all transactions to get the M value.
  - Customers with the most recent purchase, more transactions and spending more money are assigned higher values.
  - RFM helps divide customers into various categories or clusters to identify customers who are more likely to respond to promotions and for future personalization services.
  - In most of the researches, two methods are common for identifying loyal customers, one of them is in terms of demographic variables (such as age, gender, etc.) and the other is in terms of **interactive behaviours** of customers that are expressed with the so-called RFM. RFM model is proposed by Hughes in 1994 and has been used in direct marketing for several decades. This model identifies customer behaviour and represents customer behaviour characteristics by three variables: · Recency

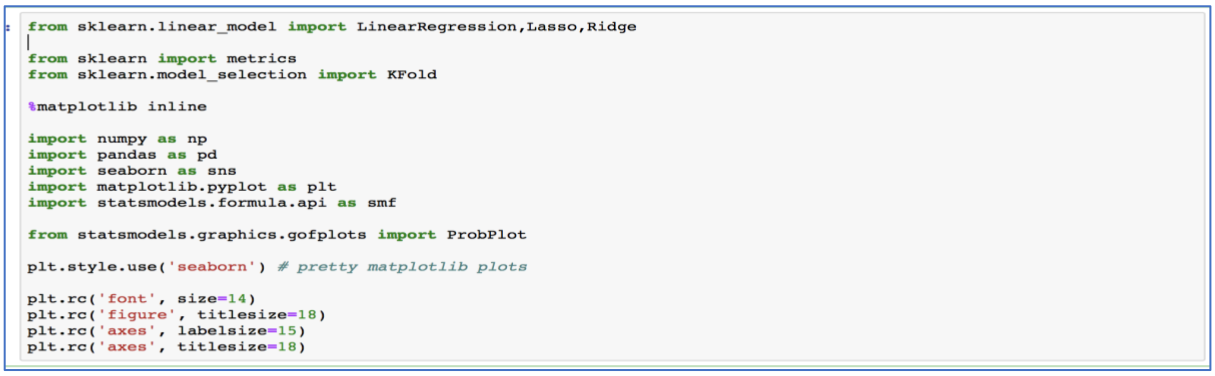
of the last purchase which refers to the interval between latest customer purchase and time analysis of customer data. · Frequency of the purchases which refers to the number of transactions in a period. · Monetary value of the purchase which refers to consumption money amount in a period.

- RFM model can be used in different areas by different people; Therefore, RFM can mean different things to different people. Classic RFM ranks each customer based on valuable against other customers and an RFM Score will be assigned to each customer.
- Basically, the RFM model applies for dividing each variable to five levels, so that 20% of customers who makes the most purchase from the company are assigned the number of 5. The next 20% will be assigned as 4, etc. finally, the database divide into 125 equal parts based on recency, frequency and monetary. Customers who have the most score are profitable.
- REF: https://www.putler.com/rfm-analysis/#11segments http://www.isr-publications.com/jmcs/307/download-developing-a-model-for-measuring-customers-loyalty-and-value-with-rfm-technique-and-clustering-algorithms

- **<u>Python Libraries and packages :</u>**
  - We use python as our choice of programming language
  - To perform modelling, EDA,plotting we use the below mentioned libraries major amongst them are SKLearn , StatsModels,Seaborn,Matplotlib,Pandas Profiling



```
In [4]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline
        import seaborn as sns
        from scipy.stats import norm
        import pandas_profiling
        from statsmodels.stats.outliers_influence import variance_inflation_factor
        pd.set_option('float_format', '{:f}'.format)
        pd.set_option('display.max_columns', None)
```



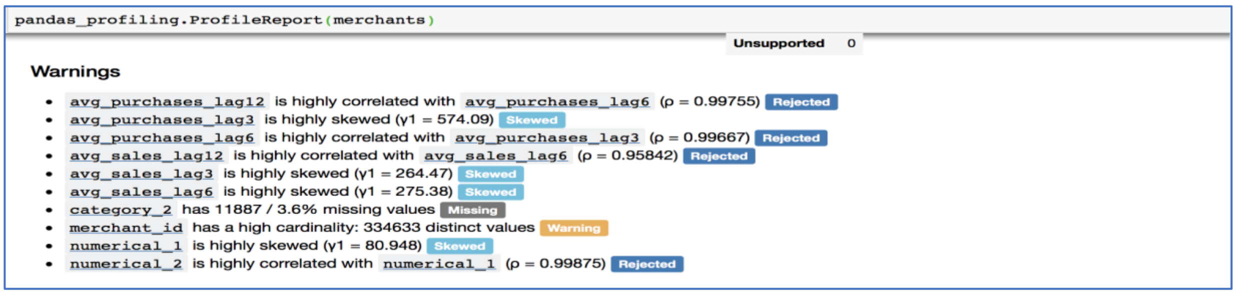
```
from sklearn.linear_model import LinearRegression,Lasso,Ridge

from sklearn import metrics
from sklearn.model_selection import KFold

%matplotlib inline

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf

from statsmodels.graphics.gofplots import ProbPlot

plt.style.use('seaborn') # pretty matplotlib plots

plt.rc('font', size=14)
plt.rc('figure', titlesize=18)
plt.rc('axes', labelsize=15)
plt.rc('axes', titlesize=18)
```

- **<u>Models \ Techniques Used :</u>**
  - Linear Regression (OLS as well)
  - Lasso
  - Ridge
  - K-Fold Validations
  - Correlation and (Vif for multicollinearity)
  - Profiling for EDA
- **<u>Transformations :</u>**
  - Standardization
  - One hot Encoding
  - PCA

- **<u>Exploratory analyses on the datasets as well as post feature engineering :</u>**

  Merchants and Historical Transactions:
  - We did profiling on Merchants dataset which gave us suggestions for duplicates, missing values ,outliers, correlations and their distribution based on which we decided things like imputation/removal of those columns or standardization.

<u>Merchants:</u> For details refer to Merchants.ipynb file.



```
pandas_profiling.ProfileReport(merchants)
```

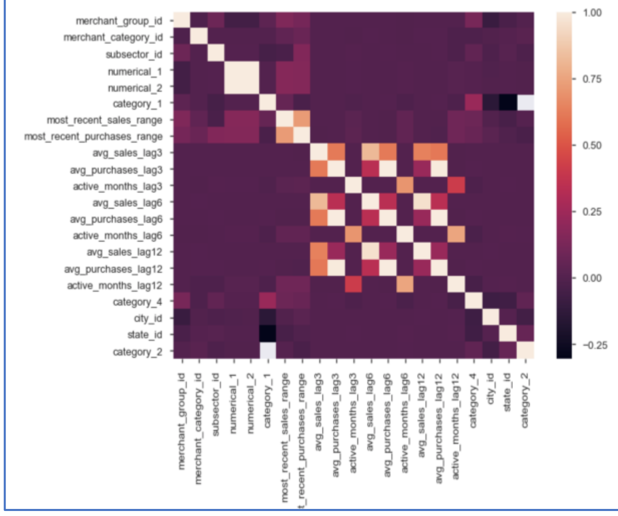
Unsupported 0

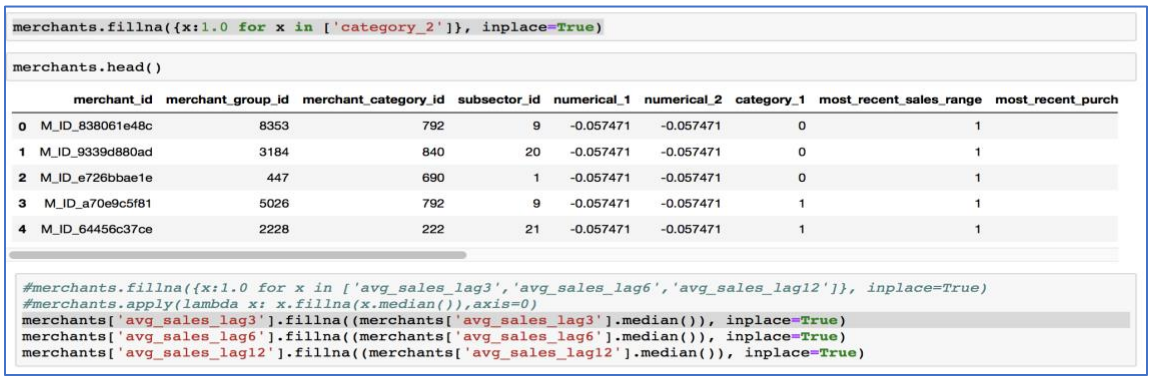
Warnings

- avg_purchases_lag12 is highly correlated with avg_purchases_lag6 (ρ = 0.99755) Rejected
- avg_purchases_lag3 is highly skewed (γ1 = 574.09) Skewed
- avg_purchases_lag6 is highly correlated with avg_purchases_lag3 (ρ = 0.99667) Rejected
- avg_sales_lag12 is highly correlated with avg_sales_lag6 (ρ = 0.95842) Rejected
- avg_sales_lag3 is highly skewed (γ1 = 264.47) Skewed
- avg_sales_lag6 is highly skewed (γ1 = 275.38) Skewed
- category_2 has 11887 / 3.6% missing values Missing
- merchant_id has a high cardinality: 334633 distinct values Warning
- numerical_1 is highly skewed (γ1 = 80.948) Skewed
- numerical_2 is highly correlated with numerical_1 (ρ = 0.99875) Rejected

```
sns.heatmap(merchants.corr())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a21e2de48>
```





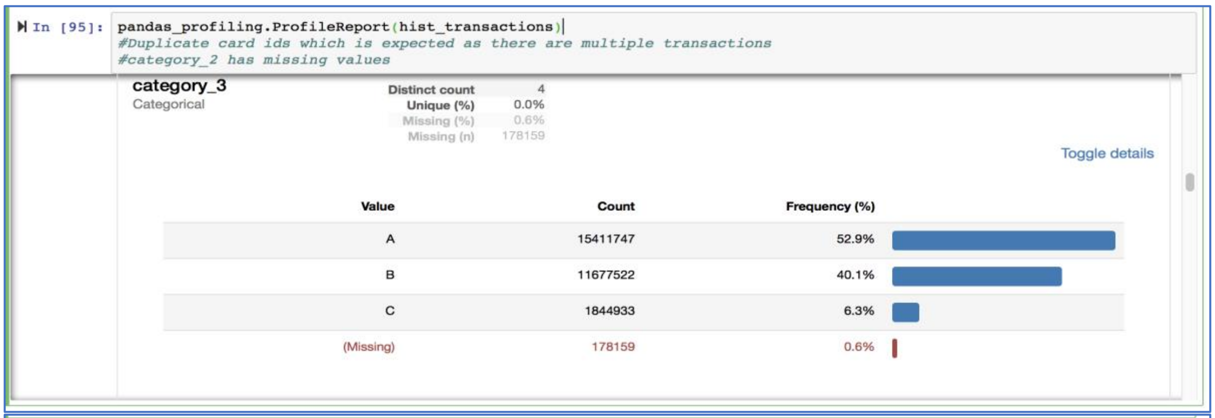
```
merchants.fillna({x:1.0 for x in ['category_2']}, inplace=True)
```

```
merchants.head()
```

	merchant_id	merchant_group_id	merchant_category_id	subsector_id	numerical_1	numerical_2	category_1	most_recent_sales_range	most_recent_purch
0	M_ID_838061e48c	8353	792	9	-0.057471	-0.057471	0	1	
1	M_ID_9339d880ad	3184	840	20	-0.057471	-0.057471	0	1	
2	M_ID_e726bbae1e	447	690	1	-0.057471	-0.057471	0	1	
3	M_ID_a70e9c5f81	5026	792	9	-0.057471	-0.057471	1	1	
4	M_ID_64456c37ce	2228	222	21	-0.057471	-0.057471	1	1	

```
#merchants.fillna({x:1.0 for x in ['avg_sales_lag3','avg_sales_lag6','avg_sales_lag12']}, inplace=True)
#merchants.apply(lambda x: x.fillna(x.median()),axis=0)
merchants['avg_sales_lag3'].fillna((merchants['avg_sales_lag3'].median()), inplace=True)
merchants['avg_sales_lag6'].fillna((merchants['avg_sales_lag6'].median()), inplace=True)
merchants['avg_sales_lag12'].fillna((merchants['avg_sales_lag12'].median()), inplace=True)
```

Historical :



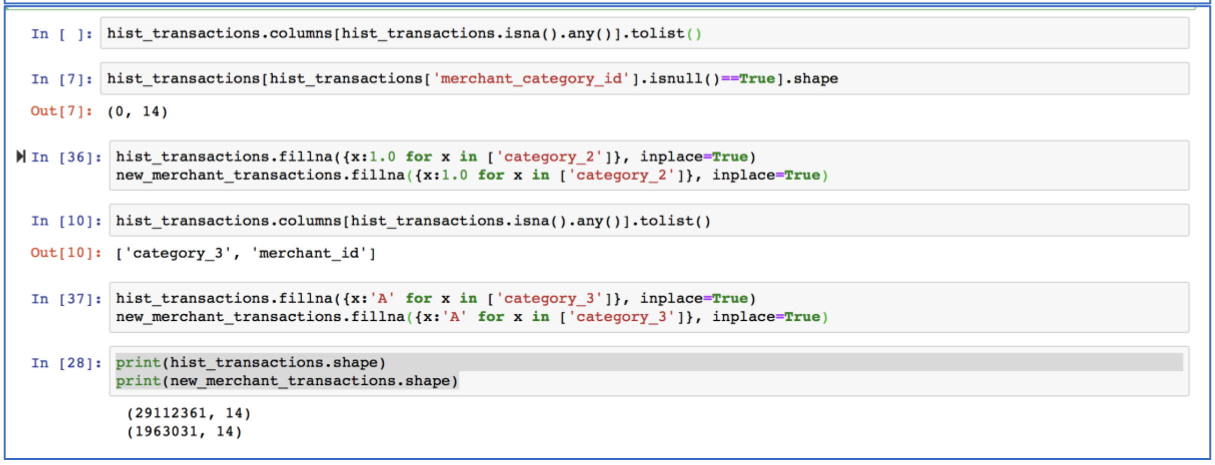
```
In [95]: pandas_profiling.ProfileReport(hist_transactions)
         #Duplicate card ids which is expected as there are multiple transactions
         #category_2 has missing values
```

category_3
Categorical

Distinct count	4
Unique (%)	0.0%
Missing (%)	0.6%
Missing (n)	178159

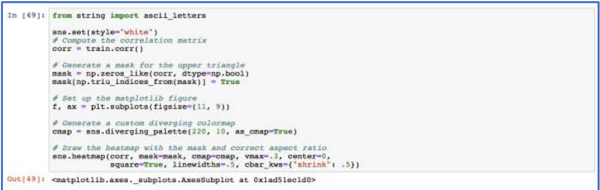
Toggle details

Value	Count	Frequency (%)
A	15411747	52.9%
B	11677522	40.1%
C	1844933	6.3%
(Missing)	178159	0.6%

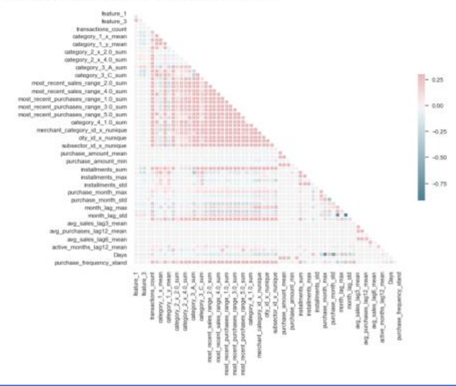


```
In [ ]: hist_transactions.columns[hist_transactions.isna().any()].tolist()

In [7]: hist_transactions[hist_transactions['merchant_category_id'].isnull()==True].shape

Out[7]: (0, 14)

In [36]: hist_transactions.fillna({x:1.0 for x in ['category_2']}, inplace=True)
         new_merchant_transactions.fillna({x:1.0 for x in ['category_2']}, inplace=True)

In [10]: hist_transactions.columns[hist_transactions.isna().any()].tolist()

Out[10]: ['category_3', 'merchant_id']

In [37]: hist_transactions.fillna({x:'A' for x in ['category_3']}, inplace=True)
         new_merchant_transactions.fillna({x:'A' for x in ['category_3']}, inplace=True)

In [28]: print(hist_transactions.shape)
         print(new_merchant_transactions.shape)

          (29112361, 14)
          (1963031, 14)
```

- **Feature Engineering(Expansion and Selection):**
    - **Merged train correlation**

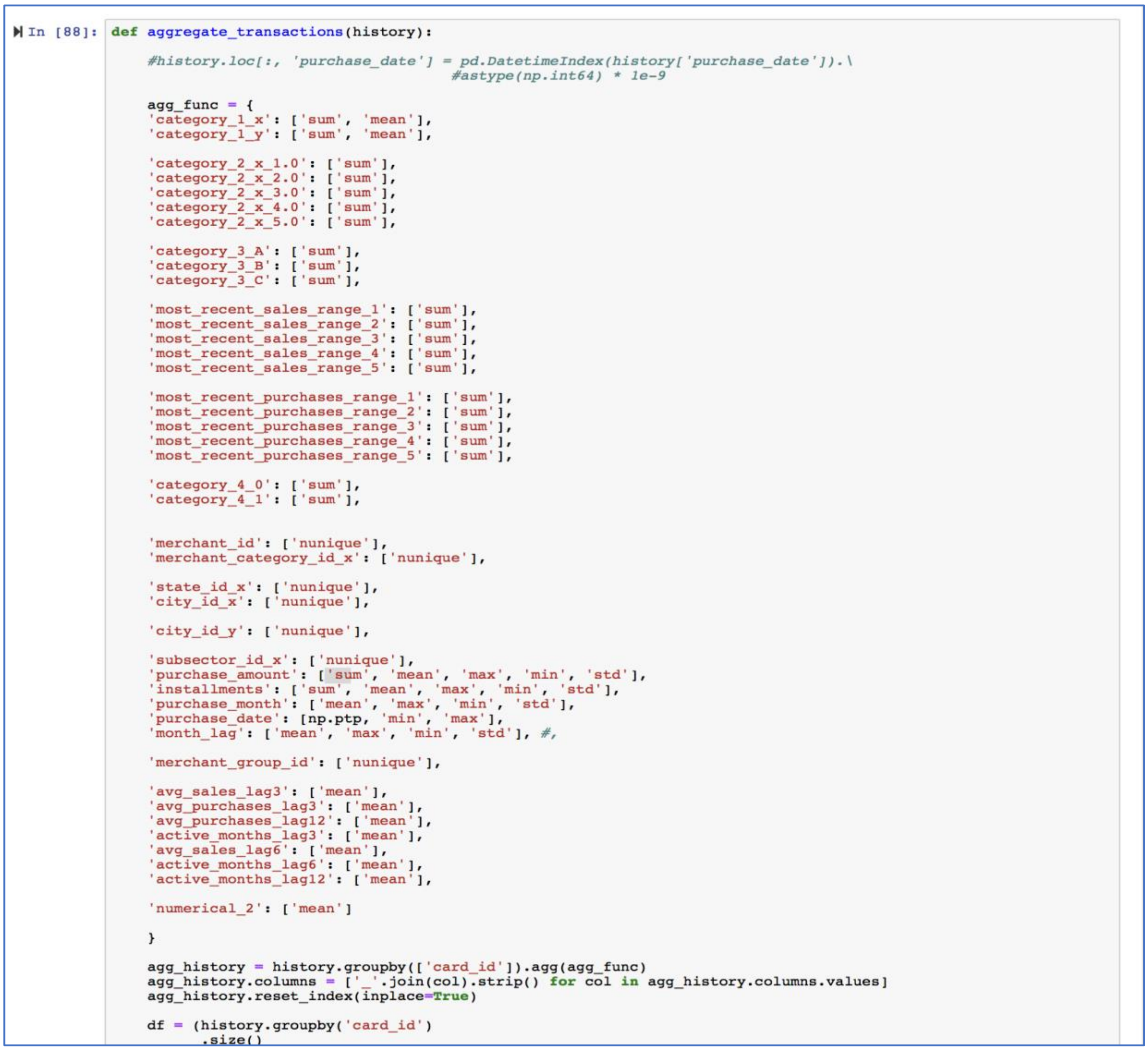


```
In [49]: from string import ascii_letters

         sns.set(style="white")
         # Compute the correlation matrix
         corr = train.corr()

         # Generate a mask for the upper triangle
         mask = np.zeros_like(corr, dtype=np.bool)
         mask[np.triu_indices_from(mask)] = True

         # Set up the matplotlib figure
         f, ax = plt.subplots(figsize=(11, 9))

         # Generate a custom diverging colormap
         cmap = sns.diverging_palette(220, 10, as_cmap=True)

         # Draw the heatmap with the mask and correct aspect ratio
         sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
                     square=True, linewidths=.5, cbar_kws={"shrink": .5})

Out[49]: <matplotlib.axes._subplots.AxesSubplot at 0x1ad51ec1d0>
```

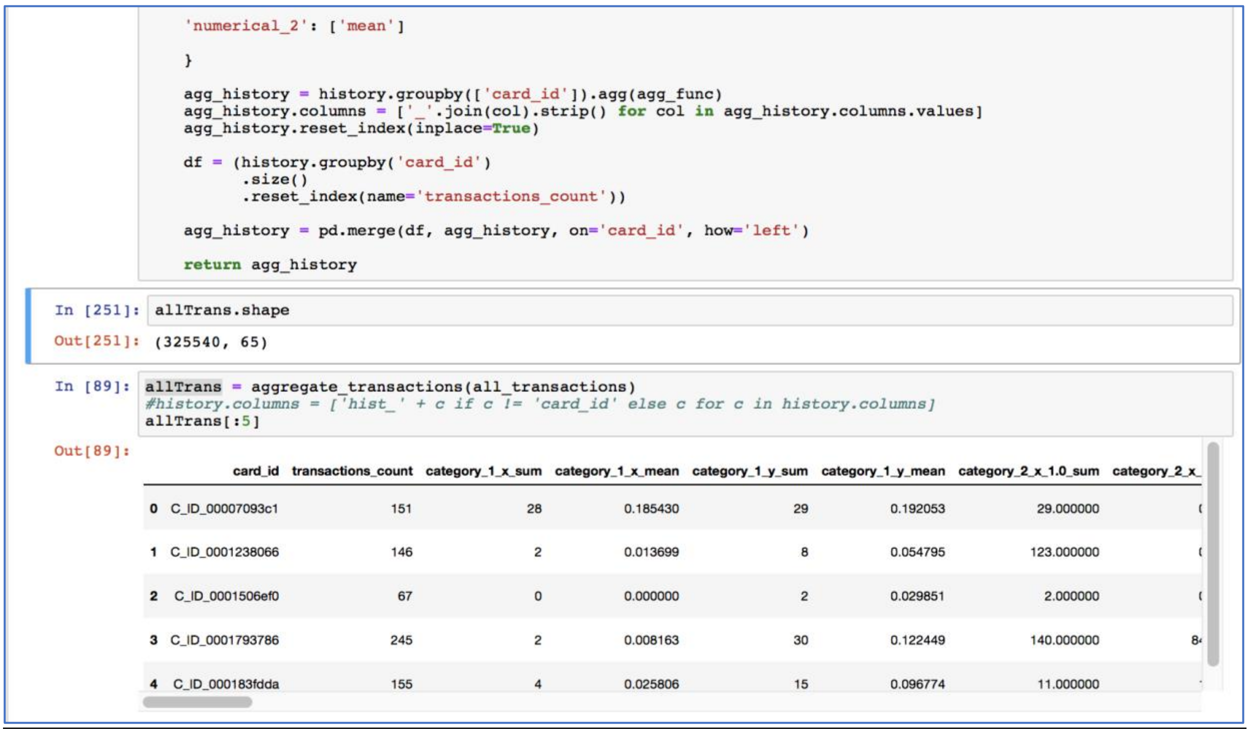


-

    - **Aggregation**



```
In [88]: def aggregate_transactions(history):

             #history.loc[:, 'purchase_date'] = pd.DatetimeIndex(history['purchase_date']).\
                                                 #astype(np.int64) * 1e-9

             agg_func = {
             'category_1_x': ['sum', 'mean'],
             'category_1_y': ['sum', 'mean'],

             'category_2_x_1.0': ['sum'],
             'category_2_x_2.0': ['sum'],
             'category_2_x_3.0': ['sum'],
             'category_2_x_4.0': ['sum'],
             'category_2_x_5.0': ['sum'],

             'category_3_A': ['sum'],
             'category_3_B': ['sum'],
             'category_3_C': ['sum'],

             'most_recent_sales_range_1': ['sum'],
             'most_recent_sales_range_2': ['sum'],
             'most_recent_sales_range_3': ['sum'],
             'most_recent_sales_range_4': ['sum'],
             'most_recent_sales_range_5': ['sum'],

             'most_recent_purchases_range_1': ['sum'],
             'most_recent_purchases_range_2': ['sum'],
             'most_recent_purchases_range_3': ['sum'],
             'most_recent_purchases_range_4': ['sum'],
             'most_recent_purchases_range_5': ['sum'],

             'category_4_0': ['sum'],
             'category_4_1': ['sum'],


             'merchant_id': ['nunique'],
             'merchant_category_id_x': ['nunique'],

             'state_id_x': ['nunique'],
             'city_id_x': ['nunique'],

             'city_id_y': ['nunique'],

             'subsector_id_x': ['nunique'],
             'purchase_amount': ['sum', 'mean', 'max', 'min', 'std'],
             'installments': ['sum', 'mean', 'max', 'min', 'std'],
             'purchase_month': ['mean', 'max', 'min', 'std'],
             'purchase_date': [np.ptp, 'min', 'max'],
             'month_lag': ['mean', 'max', 'min', 'std'], #,

             'merchant_group_id': ['nunique'],

             'avg_sales_lag3': ['mean'],
             'avg_purchases_lag3': ['mean'],
             'avg_purchases_lag12': ['mean'],
             'active_months_lag3': ['mean'],
             'avg_sales_lag6': ['mean'],
             'active_months_lag6': ['mean'],
             'active_months_lag12': ['mean'],

             'numerical_2': ['mean']

             }

             agg_history = history.groupby(['card_id']).agg(agg_func)
             agg_history.columns = ['_'.join(col).strip() for col in agg_history.columns.values]
             agg_history.reset_index(inplace=True)

             df = (history.groupby('card_id')
                   .size()
```



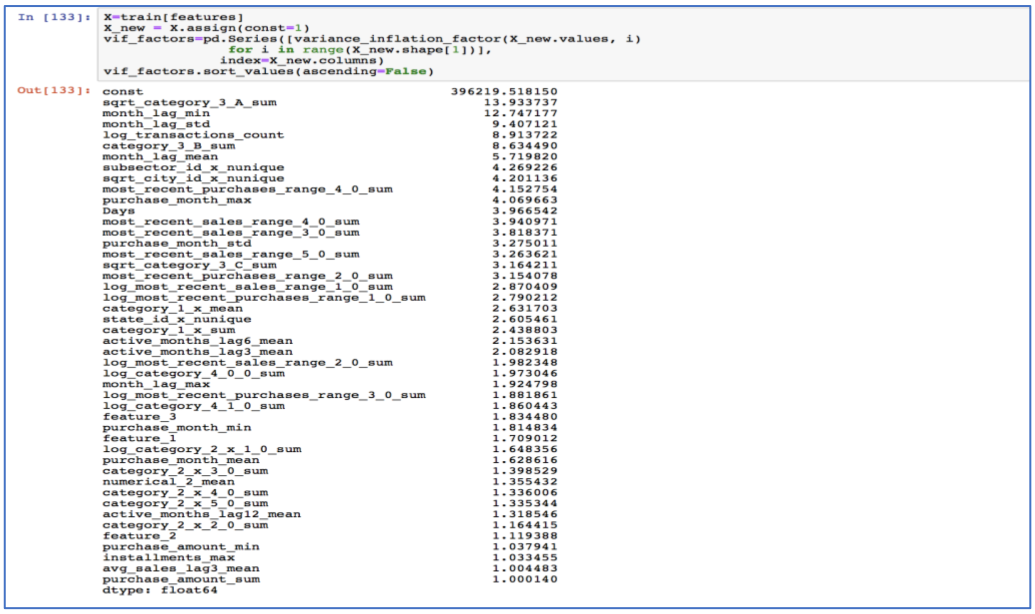
```
    'numerical_2': ['mean']

    }

    agg_history = history.groupby(['card_id']).agg(agg_func)
    agg_history.columns = ['_'.join(col).strip() for col in agg_history.columns.values]
    agg_history.reset_index(inplace=True)

    df = (history.groupby('card_id')
          .size()
          .reset_index(name='transactions_count'))

    agg_history = pd.merge(df, agg_history, on='card_id', how='left')

    return agg_history
```

```
In [251]: allTrans.shape
Out[251]: (325540, 65)
```

```
In [89]: allTrans = aggregate_transactions(all_transactions)
         #history.columns = ['hist_' + c if c != 'card_id' else c for c in history.columns]
         allTrans[:5]
Out[89]:
```

	card_id	transactions_count	category_1_x_sum	category_1_x_mean	category_1_y_sum	category_1_y_mean	category_2_x_1.0_sum	category_2_x_
0	C_ID_00007093c1	151	28	0.185430	29	0.192053	29.000000	(
1	C_ID_0001238066	146	2	0.013699	8	0.054795	123.000000	(
2	C_ID_0001506ef0	67	0	0.000000	2	0.029851	2.000000	(
3	C_ID_0001793786	245	2	0.008163	30	0.122449	140.000000	84
4	C_ID_000183fdda	155	4	0.025806	15	0.096774	11.000000	

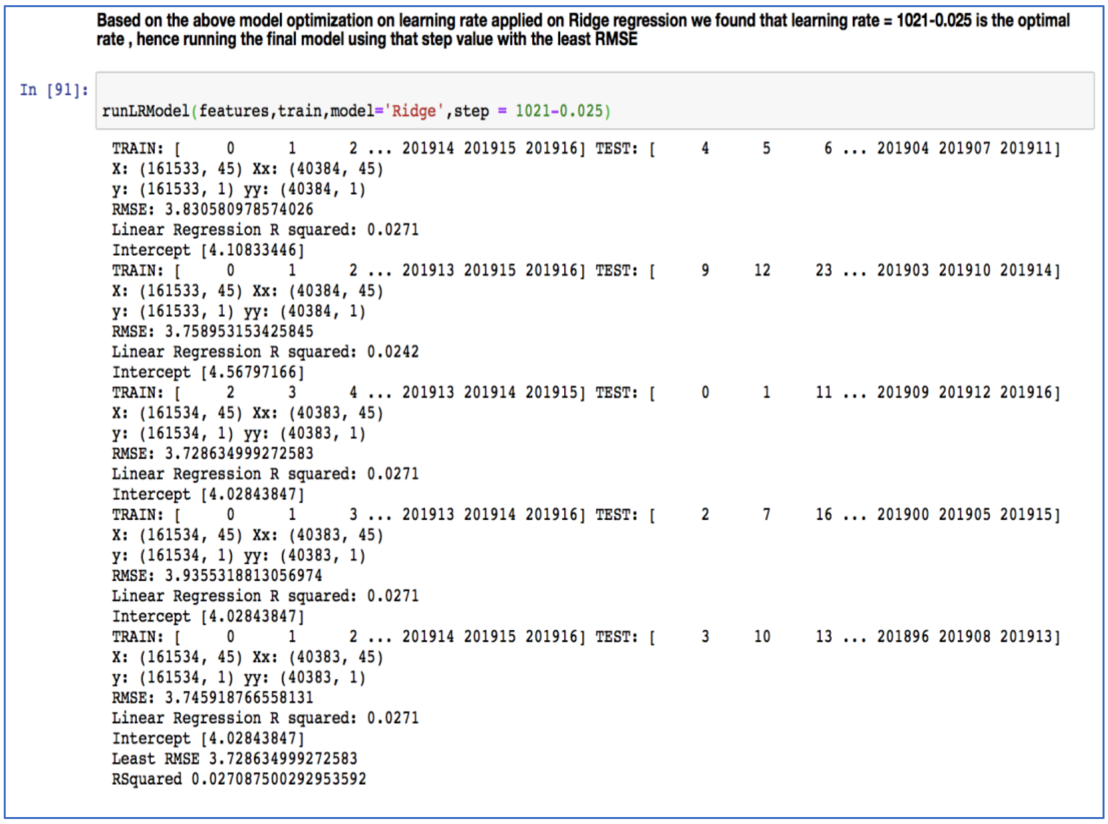
- **VIF**



```
In [133]: X=train[features]
          X_new = X.assign(const=1)
          vif_factors=pd.Series([variance_inflation_factor(X_new.values, i)
                                 for i in range(X_new.shape[1])],
                                index=X_new.columns)
          vif_factors.sort_values(ascending=False)
Out[133]: const                                          396219.518150
          sqrt_category_3_A_sum                              13.933737
          month_lag_min                                      12.747177
          month_lag_std                                       9.407121
          log_transactions_count                              8.913722
          category_3_B_sum                                    8.634490
          month_lag_mean                                      5.719820
          subsector_id_x_nunique                              4.269226
          sqrt_city_id_x_nunique                              4.201136
          most_recent_purchases_range_4_0_sum                 4.152754
          purchase_month_max                                  4.069663
          Days                                                3.966542
          most_recent_sales_range_4_0_sum                     3.940971
          most_recent_sales_range_3_0_sum                     3.818371
          purchase_month_std                                  3.275011
          most_recent_sales_range_5_0_sum                     3.263621
          sqrt_category_3_C_sum                               3.164211
          most_recent_purchases_range_2_0_sum                 3.154078
          log_most_recent_sales_range_1_0_sum                 2.870409
          log_most_recent_purchases_range_1_0_sum             2.790212
          category_1_x_mean                                   2.631703
          state_id_x_nunique                                  2.605461
          category_1_x_sum                                    2.438803
          active_months_lag6_mean                             2.153631
          active_months_lag3_mean                             2.082918
          log_most_recent_sales_range_2_0_sum                 1.982348
          log_category_4_0_0_sum                              1.973046
          month_lag_max                                       1.924798
          log_most_recent_purchases_range_3_0_sum             1.881861
          log_category_4_1_0_sum                              1.860443
          feature_3                                           1.834480
          purchase_month_min                                  1.814834
          feature_1                                           1.709012
          log_category_2_x_1_0_sum                            1.648356
          purchase_month_mean                                 1.628616
          category_2_x_3_0_sum                                1.398529
          numerical_2_mean                                    1.355432
          category_2_x_4_0_sum                                1.336006
          category_2_x_5_0_sum                                1.335344
          active_months_lag12_mean                            1.318546
          category_2_x_2_0_sum                                1.164415
          feature_2                                           1.119388
          purchase_amount_min                                 1.037941
          installments_max                                    1.033455
          avg_sales_lag3_mean                                 1.004483
          purchase_amount_sum                                 1.000140
          dtype: float64
```

- **<u>Models Executed :</u>**

We tried various flavours of Linear regression as described below :



Based on the above model optimization on learning rate applied on Ridge regression we found that learning rate = 1021-0.025 is the optimal rate , hence running the final model using that step value with the least RMSE

```
In [91]:  runLRModel(features,train,model='Ridge',step = 1021-0.025)
```

```
TRAIN: [      0      1      2 ... 201914 201915 201916] TEST: [      4      5      6 ... 201904 201907 201911]
X: (161533, 45) Xx: (40384, 45)
y: (161533, 1) yy: (40384, 1)
RMSE: 3.830580978574026
Linear Regression R squared: 0.0271
Intercept [4.10833446]
TRAIN: [      0      1      2 ... 201913 201915 201916] TEST: [      9     12     23 ... 201903 201910 201914]
X: (161533, 45) Xx: (40384, 45)
y: (161533, 1) yy: (40384, 1)
RMSE: 3.758953153425845
Linear Regression R squared: 0.0242
Intercept [4.56797166]
TRAIN: [      2      3      4 ... 201913 201914 201915] TEST: [      0      1     11 ... 201909 201912 201916]
X: (161534, 45) Xx: (40383, 45)
y: (161534, 1) yy: (40383, 1)
RMSE: 3.728634999272583
Linear Regression R squared: 0.0271
Intercept [4.02843847]
TRAIN: [      0      1      3 ... 201913 201914 201916] TEST: [      2      7     16 ... 201900 201905 201915]
X: (161534, 45) Xx: (40383, 45)
y: (161534, 1) yy: (40383, 1)
RMSE: 3.9355318813056974
Linear Regression R squared: 0.0271
Intercept [4.02843847]
TRAIN: [      0      1      2 ... 201914 201915 201916] TEST: [      3     10     13 ... 201896 201908 201913]
X: (161534, 45) Xx: (40383, 45)
y: (161534, 1) yy: (40383, 1)
RMSE: 3.745918766558131
Linear Regression R squared: 0.0271
Intercept [4.02843847]
Least RMSE 3.728634999272583
RSquared 0.027087500292953592
```

<u>Linear Regression with K-Fold :</u>

- We tried Linear Regression model based on the derived features after aggregating the train , historical , new merchant transactions and merchants features.
- Within linear regression we used KFold to make the best model selection where K= 5
- We perform two runs of this model by trying different features , one of which was on all features that we derived , and then we looked at the VIF to remove the features which had high vif values as we suspected multicollinearity for those features.

<u>OLS :</u>

- We tried OLS(Ordinary Least Squares) which is stats based model t to see its performance compared to Linear Regression which is based on machine learning.

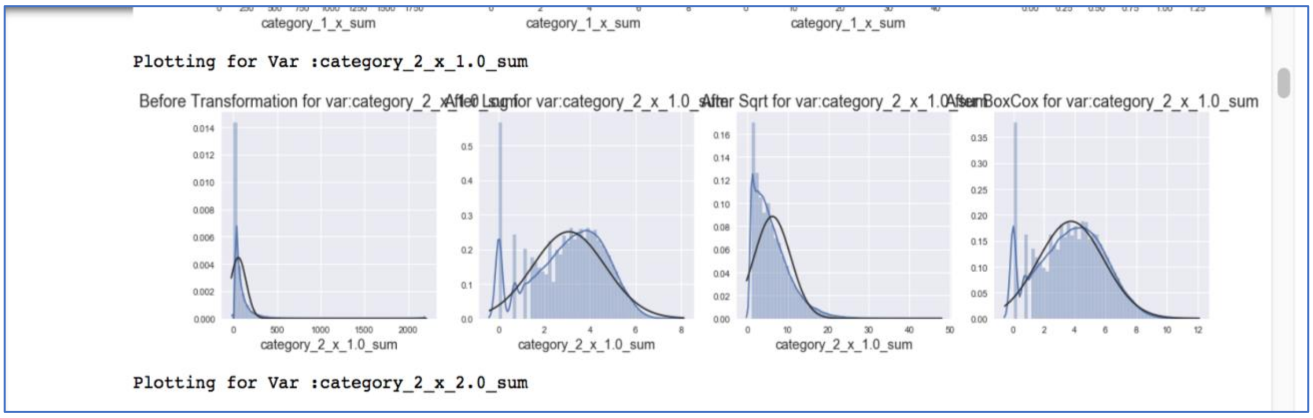
<u>Lasso:</u>

- This one is a regularized form of linear regression which performs L1 (absolute sum of coefficients) regularization

<u>Ridge:</u>
- This one is a regularized form of linear regression which performs L2 (sum of squares of coefficients) regularization
- We also tried Ridge with Feature selection as per profiling report , transformation and model optimization on learning rate

- **<u>Possible Transformations</u>** :

  - We standardized all the features and then we did PCA for better feature selection and explainability . But we found that on 99% EXPLANABILITY WE found just one principal component , and hence it did not improve our model's RMSE , hence we sticked our classic approach of backward feature elimination.
  - We transformed fields to normalize them by
    - Using Standard Scaler to normalize them
    - By using One hot encoding and binarization for categorical variables
  - We did SQRT , log and box cox- transformations to incorporate features that were normalized



```
Plotting for Var :category_2_x_1.0_sum
```

category_1_x_sum
category_1_x_sum
category_1_x_sum
Before Transformation for var:category_2_x_1.0_sum
After Log for var:category_2_x_1.0_sum
After Sqrt for var:category_2_x_1.0_sum
After BoxCox for var:category_2_x_1.0_sum
category_2_x_1.0_sum
category_2_x_1.0_sum
category_2_x_1.0_sum

```
Plotting for Var :category_2_x_2.0_sum
```

- **<u>References</u>** :
    - Class material for residual assumptions
    - Google for python syntax
    - Discussions with the team
    - https://www.kaggle.com/fabiendaniel/elo-world
    - https://etav.github.io/python/vif_factor_python.html
    - https://gist.github.com/fabianp/9396204419c7b638d38f

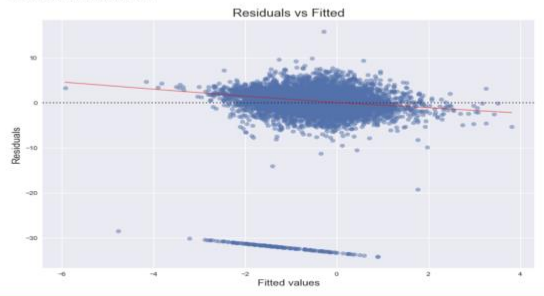
- **<u>Limitations :</u>**
    - We clearly see that this is not a perfect linear regression problem , as we tried to apply many techniques as described above even after which we see a high RMSE and a low R2 which is not acceptable and our predictions are not accurate based on that.
    - Due to kernel limitations of python , we had to stick to sample only for OLS and related residual plots.
    - We have only tried to apply linear regression methods , we intended to apply Random Forest and other Ensemble based techniques , however they are out of scope for this submission of ours

- **<u>Communicate the results and methods to a non-expert and future scope:</u>**
    - Our current model as in the current state is not suitable as of yet to be able to accurately predict the customer loyalty based on the scores as the score predictions are accurate.
    - We did cleaning of the dataset and fill in missing values so that we could take judgement on the target score based on the entire dataset.
    - We also tried to see relationships between different variables which are our predictor variables and understand if they are related and understand how they are related to our target , so that we could model them for prediction of target if new data comes.
    - Having said that we can still use some of the insights from the various features extracted and the predicted values to further see what other techniques can we use.
    - There is lot of scope for understanding the dataset better and hence coming up with features that could help us closely predict our scores. We need to see other techniques for better predictions.
    - **<u>Future Scope : We will try to incorporate , partial correlation inferences , other models to see for better variability explanation by our models</u>**

- **<u>Regression Diagnostic Plots and observations</u>** :

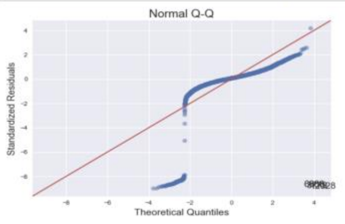
```
plot_lm_1 = plt.figure(1)
plot_lm_1.set_figheight(8)
plot_lm_1.set_figwidth(12)

plot_lm_1.axes[0] = sns.residplot(model_fitted_y, 'target', data=Z,
                          lowess=True,
                          scatter_kws={'alpha': 0.5},
                          line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8})

plot_lm_1.axes[0].set_title('Residuals vs Fitted')
plot_lm_1.axes[0].set_xlabel('Fitted values')
plot_lm_1.axes[0].set_ylabel('Residuals')
```

```
Text(0,0.5,'Residuals')
```



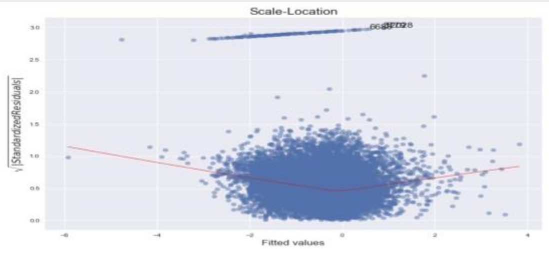
The red line in the Residuals v/s Fitted plot is not straight, hence indicating that the relationship between the predictor and the response variable target is non-linear. Linearity assumption that, red line should be straight on residual plots, and variations should be randomly scattered is not met. Couple of residuals "stands out" from the basic random pattern of residuals. This suggests that there are outliers. These same observations can be seen on the Standardized residual , the third plot .

```
In [129]: QQ = ProbPlot(model_norm_residuals)
          plot_lm_2 = QQ.qqplot(line='45', alpha=0.5, color='#4C72B0', lw=1)
          plot_lm_2.axes[0].set_title('Normal Q-Q')
          plot_lm_2.axes[0].set_xlabel('Theoretical Quantiles')
          plot_lm_2.axes[0].set_ylabel('Standardized Residuals');
          # annotations
          abs_norm_resid = np.flip(np.argsort(np.abs(model_norm_residuals)), 0)
          abs_norm_resid_top_3 = abs_norm_resid[:3]
          for r, i in enumerate(abs_norm_resid_top_3):
              plot_lm_2.axes[0].annotate(i,
                                         xy=(np.flip(QQ.theoretical_quantiles, 0)[r],
                                             model_norm_residuals[i]));
```



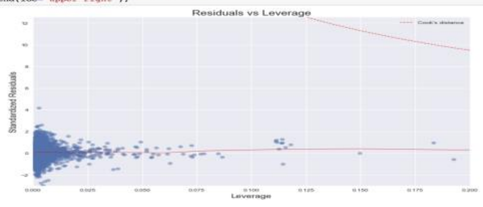
Looking further at the Normal QQ plot we see that for most cases the normality assumption is not met as the Standardized points do not follow the Theoretical quantiles on a straight diagonal line, There are also outliers that are away from the line, these outliers can also be seen in the Residual v/s fitted and other two plots. All the diagnostic plots show that there are outliers.

```
plot_lm_3 = plt.figure(3)
plot_lm_3.set_figheight(8)
plot_lm_3.set_figwidth(12)

plt.scatter(model_fitted_y, model_norm_residuals_abs_sqrt, alpha=0.5)
sns.regplot(model_fitted_y, model_norm_residuals_abs_sqrt,
            scatter=False,
            ci=False,
            lowess=True,
            line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8})

plot_lm_3.axes[0].set_title('Scale-Location')
plot_lm_3.axes[0].set_xlabel('Fitted values')
plot_lm_3.axes[0].set_ylabel('$\sqrt{|Standardized Residuals|}$');

# annotations
abs_sq_norm_resid = np.flip(np.argsort(model_norm_residuals_abs_sqrt), 0)
abs_sq_norm_resid_top_3 = abs_sq_norm_resid[:3]

for i in abs_norm_resid_top_3:
    plot_lm_3.axes[0].annotate(i,
                               xy=(model_fitted_y.iloc[[i]],
                                   model_norm_residuals_abs_sqrt[i]));
```



Same observations as that of the first residual v/s fitted plot. Clearly indicates that our model is not very good and has limitations. Also makes us think that this is not a linear regression problem

```
plot_lm_4 = plt.figure(4)
plot_lm_4.set_figheight(8)
plot_lm_4.set_figwidth(12)

plt.scatter(model_leverage, model_norm_residuals, alpha=0.5)
sns.regplot(model_leverage, model_norm_residuals,
            scatter=False,
            ci=False,
            lowess=True,
            line_kws={'color': 'red', 'lw': 1, 'alpha': 0.8})

plot_lm_4.axes[0].set_xlim(0, 0.20)
plot_lm_4.axes[0].set_ylim(-3, 12.5)
plot_lm_4.axes[0].set_title('Residuals vs Leverage')
plot_lm_4.axes[0].set_xlabel('Leverage')
plot_lm_4.axes[0].set_ylabel('Standardized Residuals')

# annotations
leverage_top_3 = np.flip(np.argsort(model_cooks), 0)[:3]

for i in leverage_top_3:
    plot_lm_4.axes[0].annotate(i,
                               xy=(model_leverage[i],
                                   model_norm_residuals[i]))

# shenanigans for cook's distance contours
def graph(formula, x_range, label=None):
    x = x_range
    y = formula(x)
    plt.plot(x, y, label=label, lw=1, ls='--', color='red')

p = len(model_fit.params) # number of model parameters

graph(lambda x: np.sqrt((0.5 * p * (1 - x)) / x),
      np.linspace(0.001, 0.200, 50),
      'Cook\'s distance') # 0.5 line
graph(lambda x: np.sqrt((1 * p * (1 - x)) / x),
      np.linspace(0.001, 0.200, 50)) # 1 line
plt.legend(loc='upper right');
```

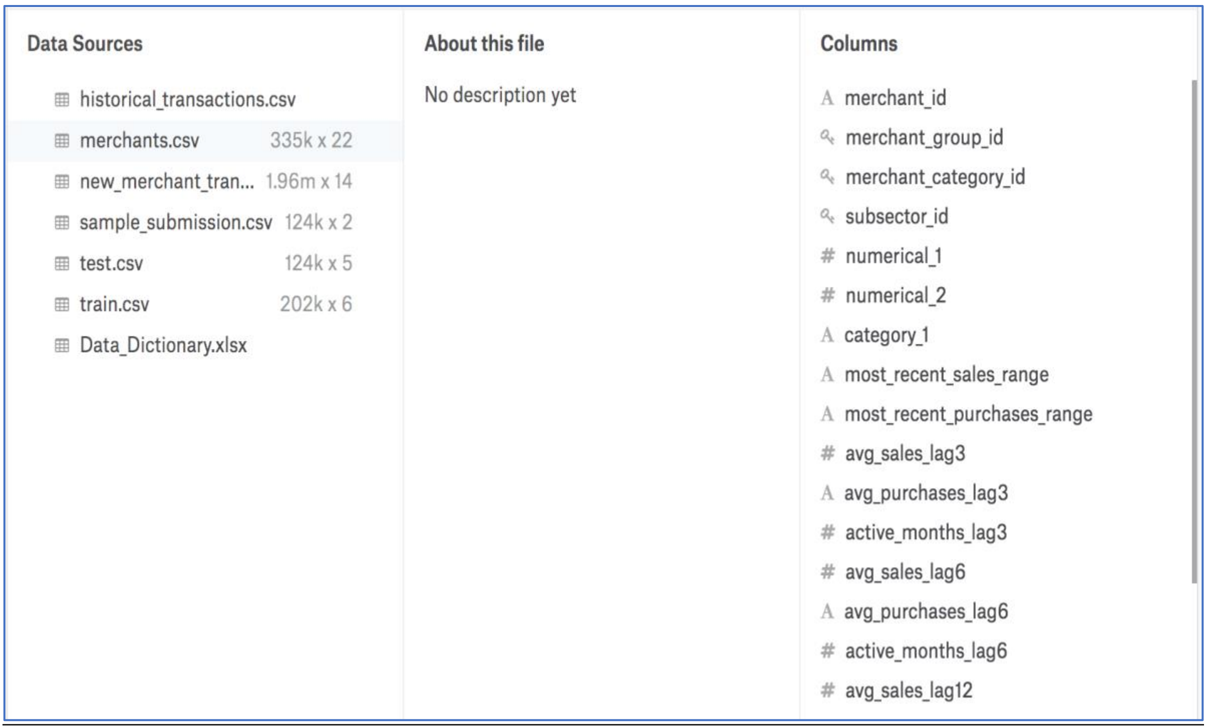


The leverage plot shows many points with a high leverage (means that this observation is an outlier due to its X value is higher than the rest of the values).

Since nothing seems to go beyond the red dotted line (the cook's distance), we conclude that there are no influential observations (which would change the slope , hence the line).

Dataset details :

- https://www.kaggle.com/c/elo-merchant-category-recommendation/data



Data Sources

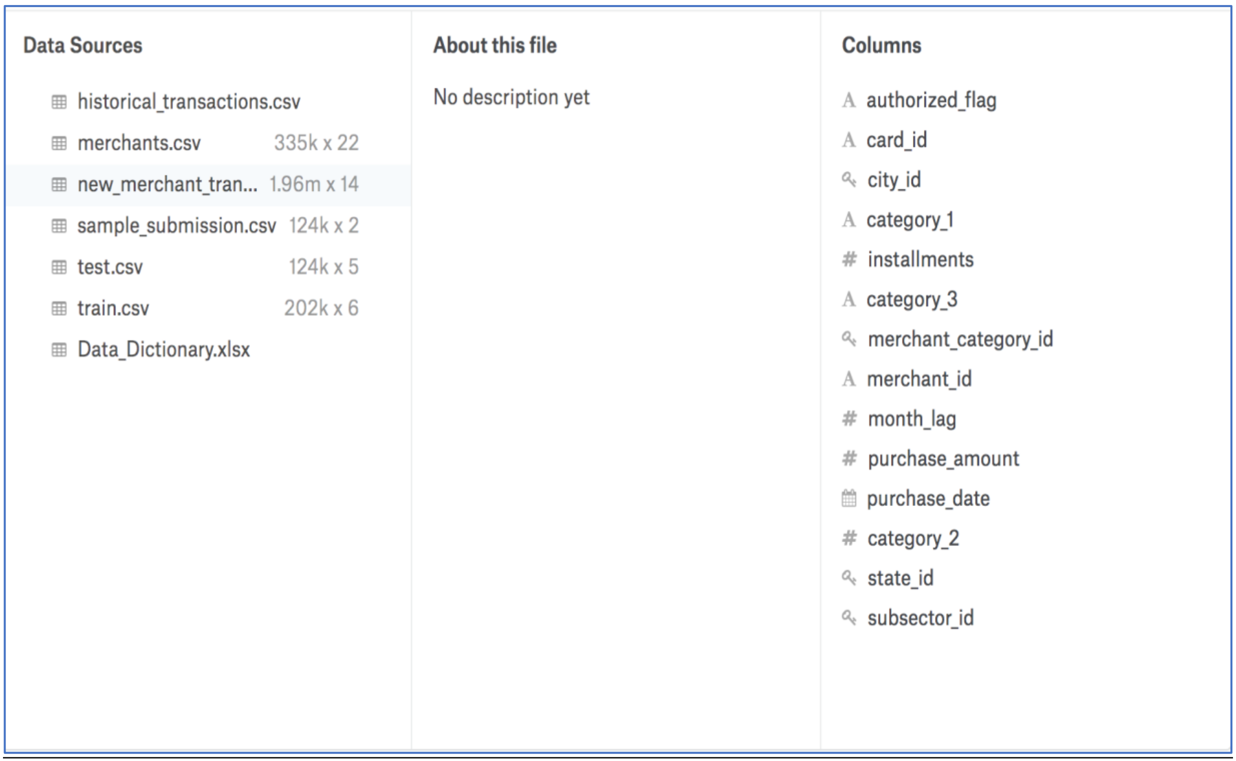
historical_transactions.csv	
merchants.csv	335k x 22
new_merchant_tran...	1.96m x 14
sample_submission.csv	124k x 2
test.csv	124k x 5
train.csv	202k x 6
Data_Dictionary.xlsx	

About this file

No description yet

Columns

- A merchant_id
- merchant_group_id
- merchant_category_id
- subsector_id
- # numerical_1
- # numerical_2
- A category_1
- A most_recent_sales_range
- A most_recent_purchases_range
- # avg_sales_lag3
- A avg_purchases_lag3
- # active_months_lag3
- # avg_sales_lag6
- A avg_purchases_lag6
- # active_months_lag6
- # avg_sales_lag12



Data Sources

historical_transactions.csv	
merchants.csv	335k x 22
new_merchant_tran...	1.96m x 14
sample_submission.csv	124k x 2
test.csv	124k x 5
train.csv	202k x 6
Data_Dictionary.xlsx	

About this file

No description yet

Columns

- A authorized_flag
- A card_id
- city_id
- A category_1
- # installments
- A category_3
- merchant_category_id
- A merchant_id
- # month_lag
- # purchase_amount
- purchase_date
- # category_2
- state_id
- subsector_id