

Elo Merchant Category Recommendation

–Help understand customer loyalty
6103 Group 4

1. Introduction

Elo, one of the largest payment brands in Brazil, has built partnerships with merchants in order to offer promotions or discounts to cardholders. However, the questions are: Do these promotions work for both consumer and merchant? Are the promotions what customers needed? Do merchants see repeat business? Personalization is key.

In this project, we aggregate merchant.csv with the new_merchant_transactions.csv and historical_transactions.csv tables and then aggregate the concatenated table to the main train table. New features are built by successive grouping on card_id, in order to recover some information. We then developed six algorithms, including linear regression, decision tree, random forest, support vector machine (SVM), K-nearest neighbors (KNN), Naive Bayes, k-means clustering, agglomerative nesting (AGNES), and density-based spatial clustering of applications with noise (DBSCAN) to predict the target: customer loyalty, in order to identify and serve the most relevant opportunities to individuals. Our goal is to improve customers' lives and help Elo reduce unwanted campaigns, to create the right experience for customers.

2. Description of the data set

train.csv and test.csv contain card_ids and information about the card itself - the first month the card was active, etc. train.csv also contains the target.

historical_transactions.csv and new_merchant_transactions.csv are designed to be joined with train.csv, test.csv, and merchants.csv. They contain information about each card's transactions. historical_transactions.csv contains up to 3 months' worth of transactions for every card at any of the provided merchant_ids. new_merchant_transactions.csv contains two months' worth of data for each card_id containing ALL purchases that card_id made at merchant_ids that were not visited in the historical data.

merchants.csv contains aggregate information for each merchant_id represented in the data set and can provide additional merchant-level information.

3. Description of the data mining and learning or cleaning algorithm or other algorithms that you used. Provide some background information on the development of the algorithm and include necessary equations and figures.

Cleaning algorithm:

Fillna: Fill NA/NaN values using the specified method.

```
DataFrame.fillna(self, value=None, method=None, axis=None, inplace=False, limit=None, downcast=None, **kwargs)[source]
```

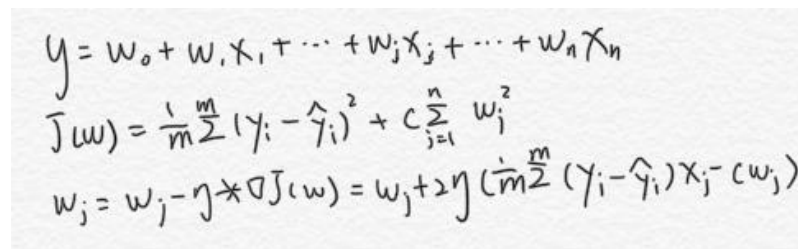
Drop outliers: In most of the cases a threshold of 3 or -3 is used i.e if the Z-score value is greater than or less than 3 or -3 respectively, that data point will be identified as outliers.

To transform the variable from continuous variable into a categorical variable, I divided intervals ranging from -12 to 12 into

- 24 groups with an interval of 1
- 6 groups with an interval of 4;
- 3 groups with an interval of 8.

We are more focus on ranging by interval 1 and ranging by interval 8.

Linear regression: one of the simplest and most commonly used statistical modeling techniques. Makes strong assumptions about the relationship between the predictor variables (x) and the response (y). Only valid for continuous outcome variables (not applicable to binary class)



The image shows handwritten mathematical formulas for linear regression. The first line is the linear model equation: $y = w_0 + w_1x_1 + \dots + w_jx_j + \dots + w_nx_n$. The second line is the cost function: $J(w) = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^n w_j^2$. The third line is the gradient descent update rule for weight w_j : $w_j = w_j - \eta \frac{\partial J(w)}{\partial w_j} = w_j + 2\eta \left(\frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i) x_{ij} - w_j \right)$.

Assumption: $y = \beta_0 + \beta_1x + \text{err}$

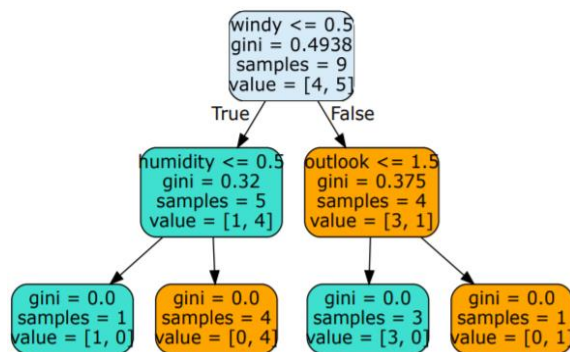
Goal: estimate β_0 and β_1 based on the available data

Final Model $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1x + \text{err}$

β_0 and β_1 are model parameters

Objective: minimize the error, the difference between our observations and the predictions made by our linear model

Decision tree: a hierarchical technique that means a series of decisions are made base on some metrics. Decision trees are Nonparametric, there are no assumptions on concerning hyperparameters or distributions. Decision trees are based on graph-based models. They are the type of Acyclic Graphs. Decision tree graphs are based on nodes and edges. These nodes and edges defined by decision rules applied to the input features.



Random forest(RF):

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual tree. Random decision forests correct for decision trees' habit of overfitting to their training set.

Support vector machine (SVM):

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features you have) with the value of each feature being the value of a particular coordinate. The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra, which is out of the scope of this introduction to SVM.

A powerful insight is that the linear SVM can be rephrased using the inner product of any two given observations, rather than the observations themselves. The inner product between two vectors is the sum of the multiplication of each pair of input values

K-nearest neighbor (KNN):

K-Nearest Neighbors is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection.

It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data).

Naive Bayes:

Naive Bayes is a simple classification technique that relies on conditional probability, and predicts the most probable class given a set of inputs. It is often used as a baseline for more complex models. Naive Bayes Classifiers are extremely fast and surprisingly

accurate given their “naive assumptions”. Naive Bayes is a simple technique for predicting the most probable class/label given a set of features/inputs.

K-Means Clustering:

K-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriority. The main idea is to define k centers, one for each cluster. These centers should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other.

Agglomerative clustering:

Agglomerative clustering: It's also known as AGNES (Agglomerative Nesting). It works in a bottom-up manner. That is, each object is initially considered as a single-element cluster (leaf). At each step of the algorithm, the two clusters that are the most similar are combined into a new bigger cluster (nodes). This procedure is iterated until all points are member of just one single big cluster (root) (see figure below). The result is a tree which can be plotted as a dendrogram.

DBSCAN

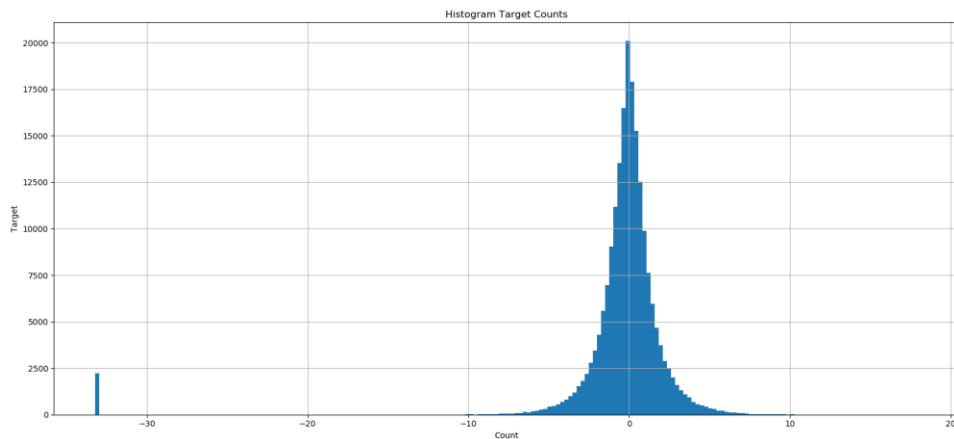
Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a very popular density based data clustering algorithm commonly used in data mining and machine learning. DBSCAN clusters the data points to separate the areas of high density with areas of low density. It also marks data points as outliers that are in the low density regions. The clusters formed can be of varying shapes based on the density of data points.

4. Experimental setup. Describe how you are going to use the data to clean and preprocess. Explain how you will implement the data mining technique in the chosen software and how you will judge the performance. Write a complete report with a theoretical description and verify this mathematical concept by applying it with actual data. Provide enough information about the codes that you have written. Write your codes in separate subroutines and call the functions if needed?. Explain each subroutine.

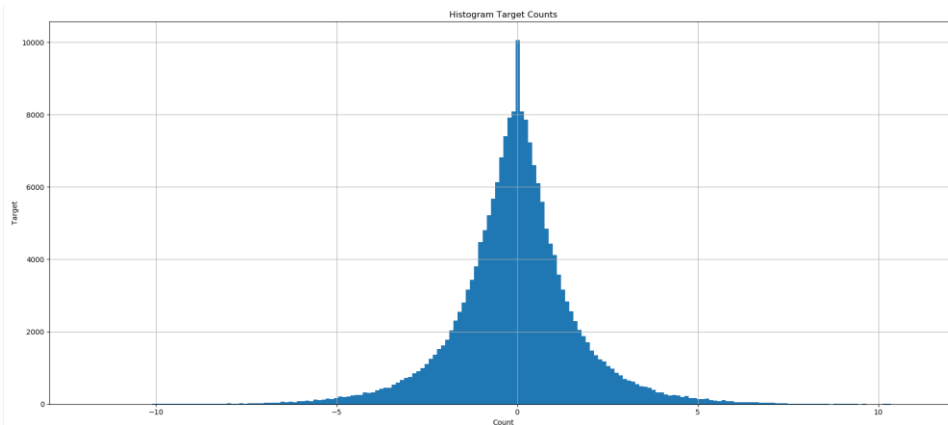
4.1 Preprocessing and feature engineering

- 1) Use function of 'missing_values_table' to check missing values of all the tables;
- 2) Check unique values of 'category_2', 'category_3' and 'merchant_id' in historical_transactions and new_merchant_transactions and fill missing values with values different from unique values existed in those columns.
- 3) Calculated z-score of target and drop outliers whose Z-score value is greater than or less than 3 or -3 respectively.

Histogram of target with outliers



Histogram of target without outliers



- 4) Get dummies of 'feature_1', 'feature_2' and 'feature_3' in historical_transactions and new_merchant_transactions, train and test.
- 5) Define functions that aggregate the info contained in tables. The first function aggregates the function by grouping on card_id; the second function first aggregates on the two variables card_id and month_lag. Then a second grouping is performed to aggregate over time.
- 6) Fill missing values after feature engineering with 0
- 7) Merge all the dataframes and then write the merged df to a .csv file

4.2 Models

4.2.1 Linear Regression

- 1) Split the data into features X and target y
- 2) Using train_test_split from sklearn.model_selection, divide the data into training and testing (with test_size=0.3 and random_state = 0)

- 3) Use a self-defined linear regression to fit a linear model.
- 4) Use GridSearchCV from sklearn.model_selection to tune hyperparameters such as learning rate and constant penalty on coefficients, with scoring='neg_mean_squared_error', n_jobs=-1, iid=False, cv=KFold(n_splits=10, random_state=0), return_train_score=True.

4.2.2 Decision Tree

- 1) As the target in the dataset is continuous, we converted the target to categories using: bins = np.arange(-12.5, 12.5, 1)

names = np.arange(-12, 12, 1)

data['new_target'] = pd.cut(data['target'], bins, labels=names)

- 2) Using LabelEncoder() to transfer the target.
- 3) Using StandardScaler() to standardize X_train and X_test
- 4) Using train_test_split from sklearn.model_selection, divide the data into training and testing (with test_size=0.3 and random_state = 0)
- 5) Use GridSearchCV from sklearn.model_selection to tune hyperparameters such as max_depth, min_samples_leaf and min_samples_split
- 6) Train a decision tree with criterion as gini and best hyperparameter selected using X_train and y_train.
- 7) Make predictions y_pred using the trained model and X_test
- 8) Calculate accuracy and mean_square_error using y_pred and y_test.
- 9) Get confusion matrix
- 10) Display decision tree.

4.2.3 Random forest

- 1) Import dataset
- 2) Split dataset
 - a) As the target in the dataset is continuous, we converted the target to categories using:

bins = np.arange(-12.5, 12.5, 1)

names = np.arange(-12, 12, 1)
 - b) As the target in the dataset is continuous, we converted the target to categories using:

```
bins = np.arange(-12.5, 12.5, 1)
names = np.arange('[-12, -4)', '[-4, 4)', '[4, 12)']
data['new_target'] = pd.cut(data['target'], bins, labels=names)
```

- 3) Perform training with random forest with all columns
- 4) Plot feature importance
- 5) select features to perform training with random forest with k columns
- 6) perform training with random forest with k columns
- 7) make predictions
- 8) calculate metrics gini model
- 9) confusion matrix for gini model
- 10) calculate metrics entropy model
- 11) Confusion matrix for entropy model

4.2.4 Naive Bayes

- 1) Import packages
- 2) Split the dataset
 - a) Split dataset
 - i) As the target in the dataset is continuous, we converted the target to categories using:

```
bins = np.arange(-12.5, 12.5, 1)
```

```
names = np.arange(-12, 12, 1)
```

- ii) As the target in the dataset is continuous, we converted the target to categories using:

```
bins = np.arange(-12.5, 12.5, 1)
```

```
names = np.arange('[-12, -4)', '[-4, 4)', '[4, 12)')
```

- 3) Perform training
- 4) Make predictions
- 5) Calculate metrics
- 6) Confusion matrix

4.2.5 KNN

- 1) Import packages and dataset
- 2) Data preprocessing
- 3) Split the dataset into train and test
 - a) Split dataset

- i) As the target in the dataset is continuous, we converted the target to categories using:

```
bins = np.arange(-12.5, 12.5, 1)
```

```
names = np.arange(-12, 12, 1)
```

- ii) As the target in the dataset is continuous, we converted the target to categories using:

```
bins = np.arange(-12.5, 12.5, 1)
```

```
names = np.arange(['-12, -4'], ['-4, 4'], ['4, 12'])
```

- 4) Standardize the data
- 5) Perform training
- 6) Make predictions
- 7) Calculate metrics
- 8) Plot confusion matrix

4.2.5.1 K-means

- 1) Import key packages:
from sklearn.cluster import KMeans
- 2) Choose the X and Y:
X = data[["purchase_amount_count_std", "auth_purchase_month_std"]]
y = data['new_target']
- 3) Use K-means classifier: estimator = KMeans(n_clusters=3)
- 4) Set predict_label and plot the result:
x0 = X[label_pred == 0]
x1 = X[label_pred == 1]
x2 = X[label_pred == 2]
plt.scatter(x0.iloc[:, 0], x0.iloc[:, 1], c="red", marker='o', label='label0')
plt.scatter(x1.iloc[:, 0], x1.iloc[:, 1], c="green", marker='*', label='label1')
plt.scatter(x2.iloc[:, 0], x2.iloc[:, 1], c="blue", marker='+', label='label2')

4.2.6 AGNES

- 1) Import key packages:
from sklearn.cluster import AgglomerativeClustering
- 2) Choose X and Y:
X = data[["month_lag_std", "auth_purchase_month_std"]]
y = data['new_target']
- 3) Use AGNES classifier:
clustering = AgglomerativeClustering(linkage='ward', n_clusters=3)
- 4) Confusion matrix:
print(confusion_matrix(data.y, clustering.labels_))
- 5) Set predict_label and plot the result:


```

x0 = X[label_pred == 0]
x1 = X[label_pred == 1]
x2 = X[label_pred == 2]
plt.scatter(x0.iloc[:, 0], x0.iloc[:, 1], c="red", marker='o', label='label0')
plt.scatter(x1.iloc[:, 0], x1.iloc[:, 1], c="green", marker='*', label='label1')
plt.scatter(x2.iloc[:, 0], x2.iloc[:, 1], c="blue", marker='+', label='label2')

```

4.2.7 DBSCAN

- 1) Import key packages:
from sklearn.cluster import DBSCAN
- 2) Choose X and Y:
X = data[["purchase_amount_count_std", "auth_purchase_month_std"]]
y = data['new_target']
- 3) Use DBSCAN classifier:
dbscan = DBSCAN(eps=0.4, min_samples=9)
- 4) Set predict_label and plot the result:
x0 = X[label_pred == 0]
x1 = X[label_pred == -1]
x2 = X[label_pred == 1]
plt.scatter(x0.iloc[:, 0], x0.iloc[:, 1], c="red", marker='o', label='label0')
plt.scatter(x1.iloc[:, 0], x1.iloc[:, 1], c="green", marker='*', label='label-1')
plt.scatter(x2.iloc[:, 0], x2.iloc[:, 1], c="blue", marker='+', label='label1')

4.2.8 SVM

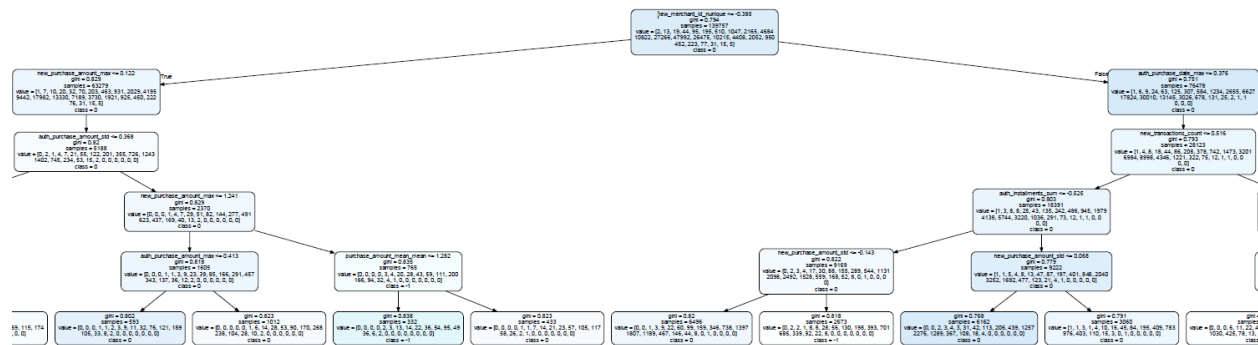
- 1) Import key packages:
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
- 2) Choose X and Y:
X = data[["month_lag_std", "auth_purchase_month_std"]]
y = data['new_target']
- 3) Split the data:
X_train, X_val, y_train, y_val =
train_test_split(X, y, test_size=0.3, random_state=13)
- 4) Use SVC classifier:
clf = SVC(C=6, kernel='rbf')
- 5) Plot the result and show the accuracy
- 6) Calculate the accuracy with samples

5. Results. Describe the results of your experiments, using figures and tables wherever possible. Include all results (including all figures and tables) in the main body of the report, not in appendices. Provide an explanation of each figure and table that you include. Your discussions in this section will be the most important part of the report.

5.1 Linear Regression

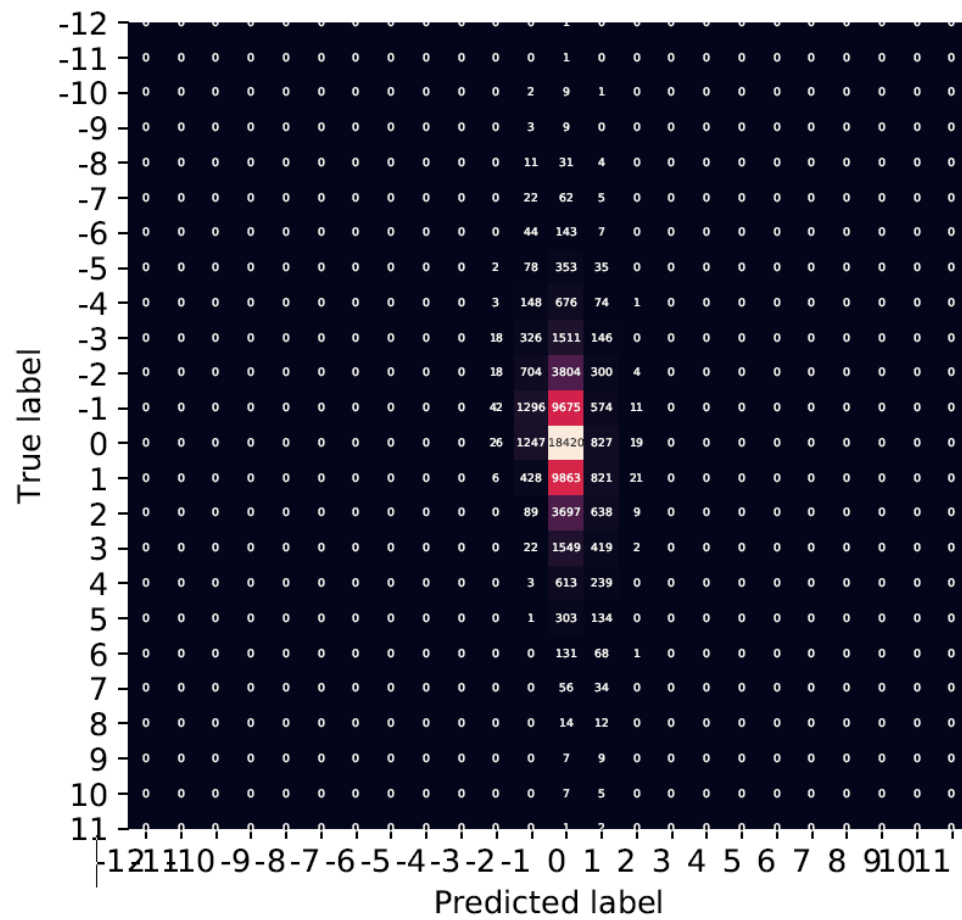
Best_score (*neg_mean_squared_error*): -0.8982704791215965; best_params: {'estimator__eta': 0.1}; Mean_squared_error: 2.65; r2_score: 0.09

Best hyperparameters are 'max_depth': 6, 'min_samples_leaf': 1, 'min_samples_split': 2



Accuracy: 34.33 and Mean_square_error: 2.87

Confusion matrix:



Classification Report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	1
1	0.00	0.00	0.00	1
2	0.00	0.00	0.00	12
3	0.00	0.00	0.00	12
4	0.00	0.00	0.00	46
5	0.00	0.00	0.00	89
6	0.00	0.00	0.00	194
7	0.00	0.00	0.00	468
8	0.00	0.00	0.00	902
9	0.00	0.00	0.00	2001
10	0.16	0.00	0.01	4830
11	0.29	0.11	0.16	11598
12	0.36	0.90	0.52	20539
13	0.19	0.07	0.11	11139
14	0.13	0.00	0.00	4433
15	0.00	0.00	0.00	1992
16	0.00	0.00	0.00	855
17	0.00	0.00	0.00	438
18	0.00	0.00	0.00	200
19	0.00	0.00	0.00	90
20	0.00	0.00	0.00	26
21	0.00	0.00	0.00	16
22	0.00	0.00	0.00	12
23	0.00	0.00	0.00	3

- o Ranging by interval 1
- ### Classification of All features

```

Classification Report:
/Users/lancy/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.py:135: FutureWarning:
no predicted samples.
'precision', 'predicted', average, warn_for)

```

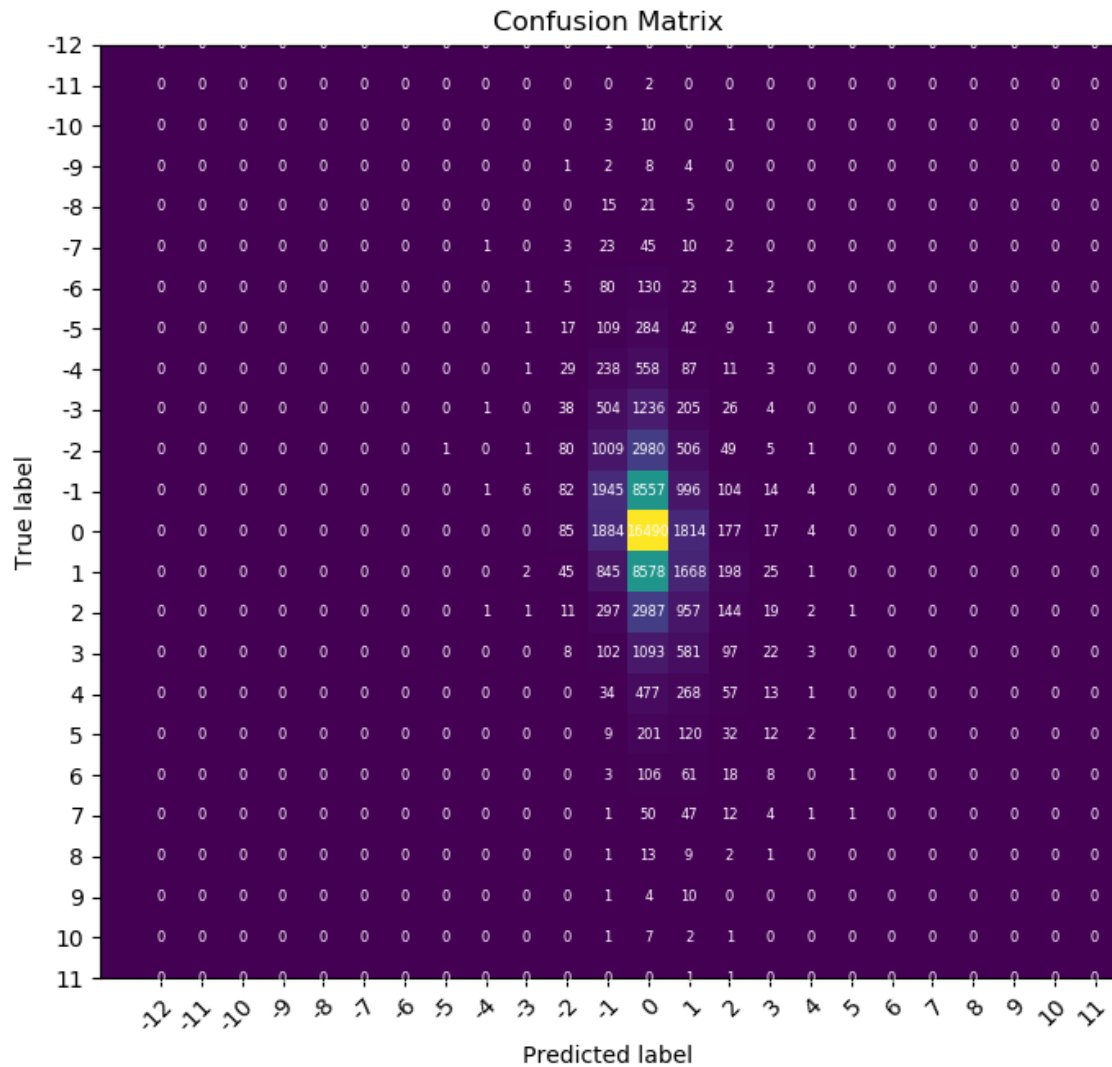
	precision	recall	f1-score	support
-12	0.00	0.00	0.00	1
-11	0.00	0.00	0.00	2
-10	0.00	0.00	0.00	14
-9	0.00	0.00	0.00	15
-8	0.00	0.00	0.00	41
-7	0.00	0.00	0.00	84
-6	0.00	0.00	0.00	242
-5	0.00	0.00	0.00	463
-4	0.00	0.00	0.00	927
-3	0.11	0.00	0.00	2014
-2	0.17	0.02	0.03	4632
-1	0.27	0.16	0.20	11709
0	0.38	0.81	0.51	20471
1	0.23	0.15	0.18	11362
2	0.15	0.03	0.05	4420
3	0.13	0.01	0.02	1906
4	0.05	0.00	0.00	850
5	0.50	0.00	0.01	377
6	0.00	0.00	0.00	197
7	0.00	0.00	0.00	116
8	0.00	0.00	0.00	26
9	0.00	0.00	0.00	15
10	0.00	0.00	0.00	11
11	0.00	0.00	0.00	2
accuracy			0.34	59897
macro avg	0.08	0.05	0.04	59897
weighted avg	0.26	0.34	0.26	59897

Accuracy : 33.93158255004424

Mean_squared_error: 2.8938678064009884

Accuracy is 33.93 and MSE is 2.89

Confusion Matrix:



We can see that target value 0 is the highest number of correct predictions.

- o Ranging by interval 8
RF all features classification report

Results Using All Features:

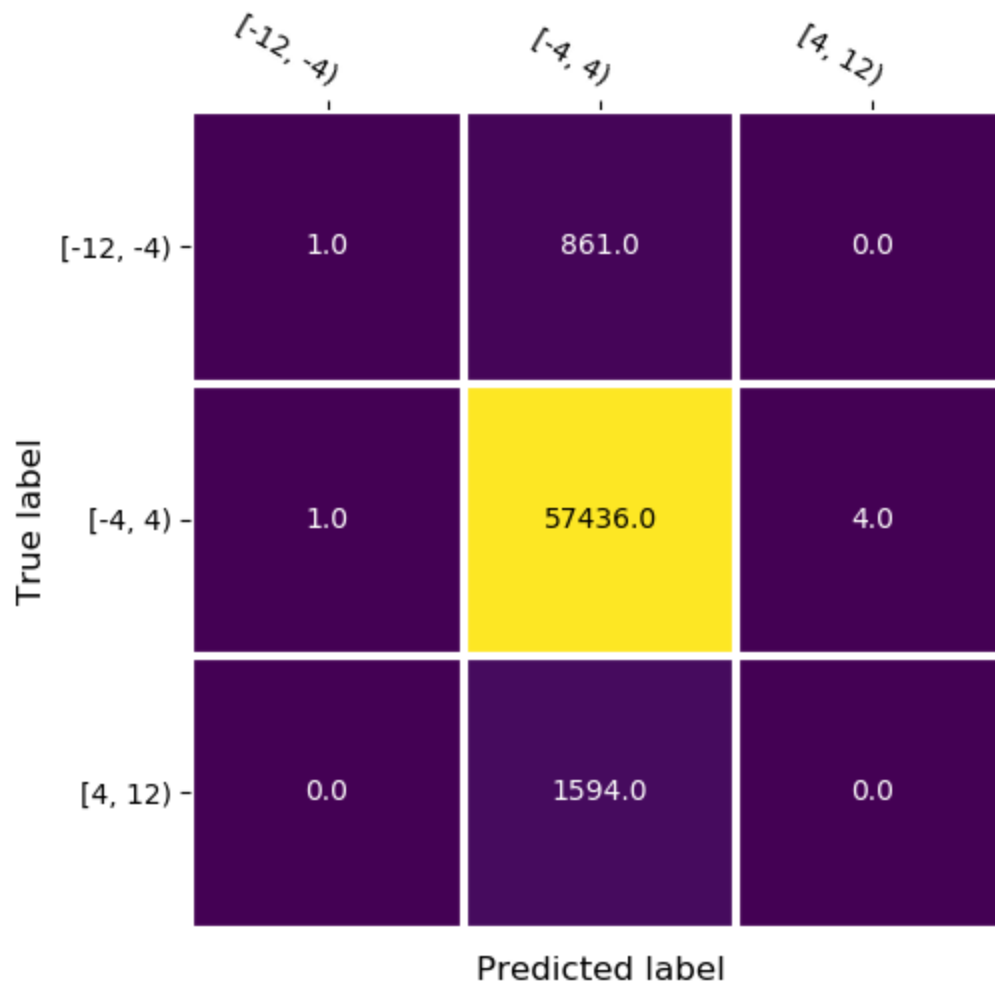
Classification Report:

	precision	recall	f1-score	support
[-12, -4)	0.50	0.00	0.00	862
[-4, 4)	0.96	1.00	0.98	57441
[4, 12)	0.50	0.00	0.00	1594
accuracy			0.96	59897
macro avg	0.65	0.33	0.33	59897
weighted avg	0.94	0.96	0.94	59897

Accuracy : 95.89962769420839

Accuracy is 95.89. We improve our model

Confusion Matrix



We can see that target value between $[-4, 4)$ is the highest number of correct predictions.

RF K features classification report

Results Using K features:

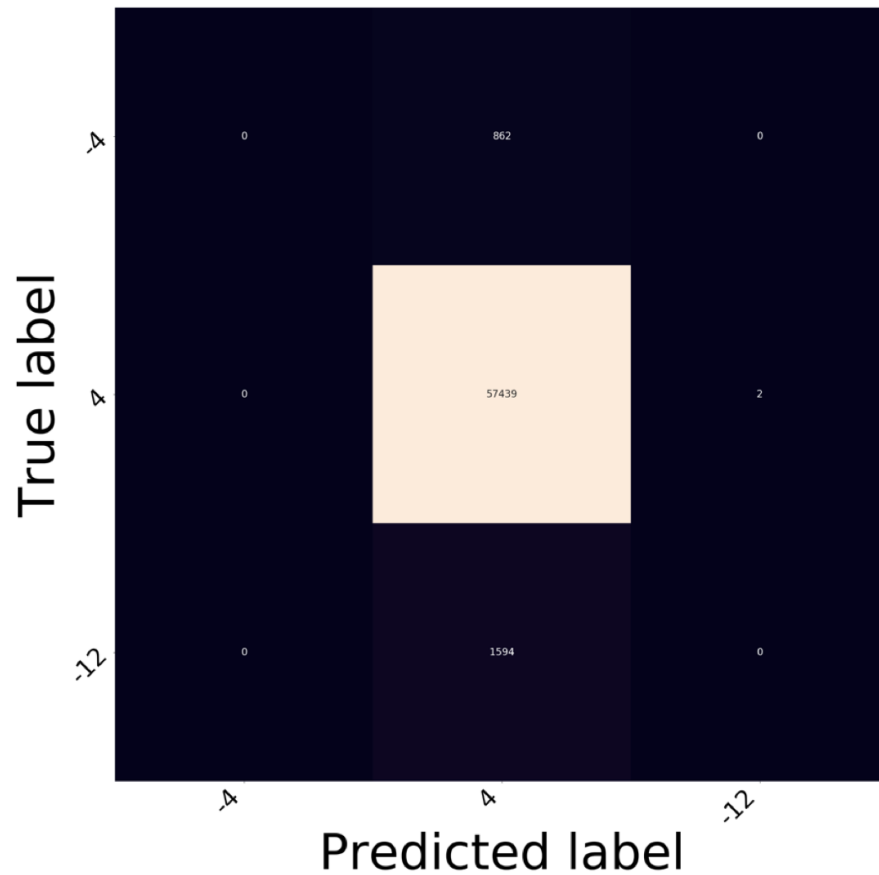
Classification Report:

	precision	recall	f1-score	support
[-12, -4)	0.00	0.00	0.00	862
[-4, 4)	0.96	1.00	0.98	57441
[4, 12)	0.33	0.00	0.00	1594
accuracy			0.96	59897
macro avg	0.43	0.33	0.33	59897
weighted avg	0.93	0.96	0.94	59897

Accuracy : 95.89795816151059

Accuracy is 95.90.

Confusion Matrix:



We can see that target value [-4,4) is the highest number of correct predictions.

5.4 Naive Bayes

- o Ranging by interval 1

Naive Bayes classification report

Cleaning algorithm:

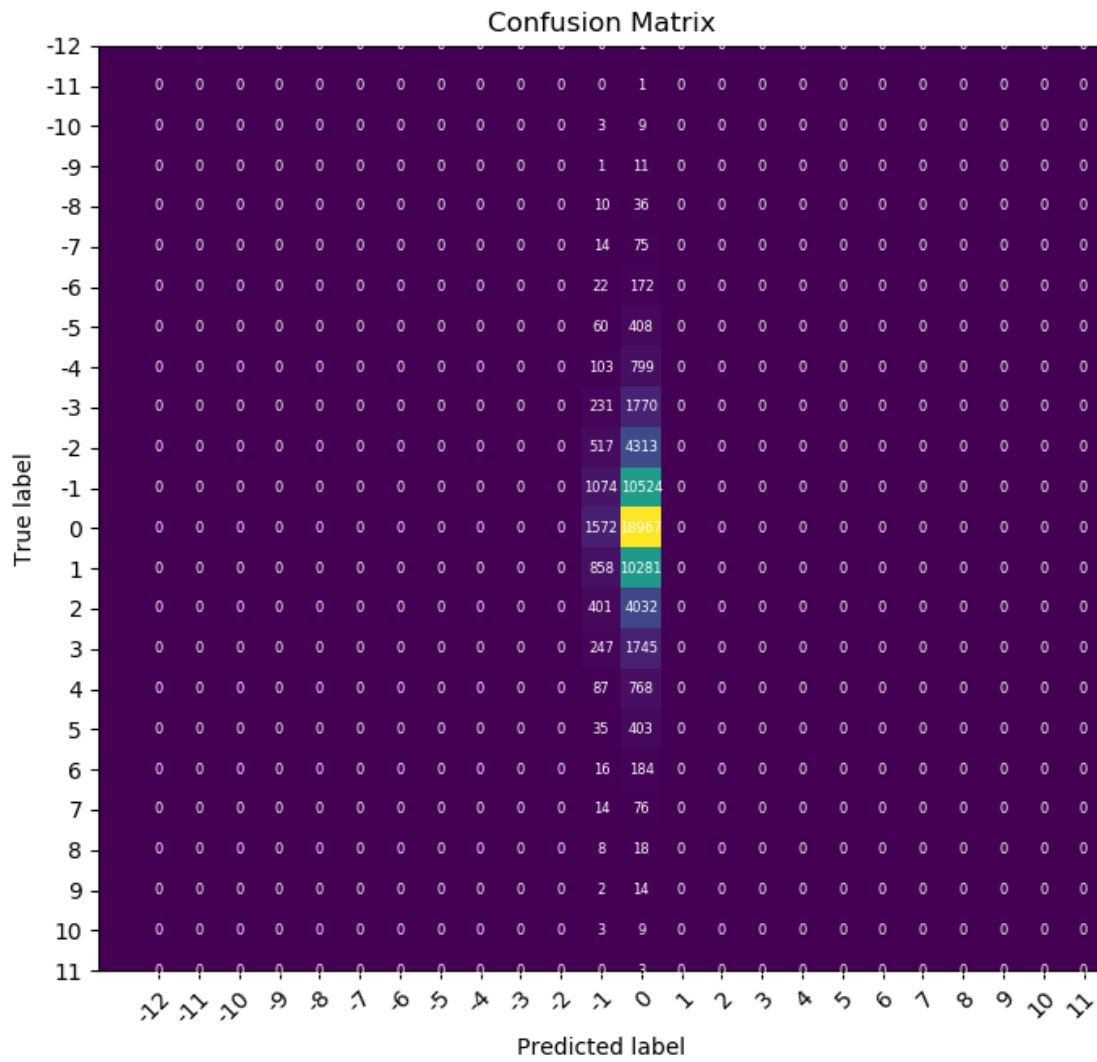
Fillna: Fill NA/NaN values using the specified method.

DataFrame.fillna(self, value=None, method=None, axis=None, inplace=False, limit=None, downcast=None, **kwargs)[source]

Drop outliers

Accuracy is 33.45, mean squared error is 3.06.

Confusion Matrix



We can see that target value 0 is the highest number of correct predictions.

- o Ranging by interval 8
Naive Bayes classification report

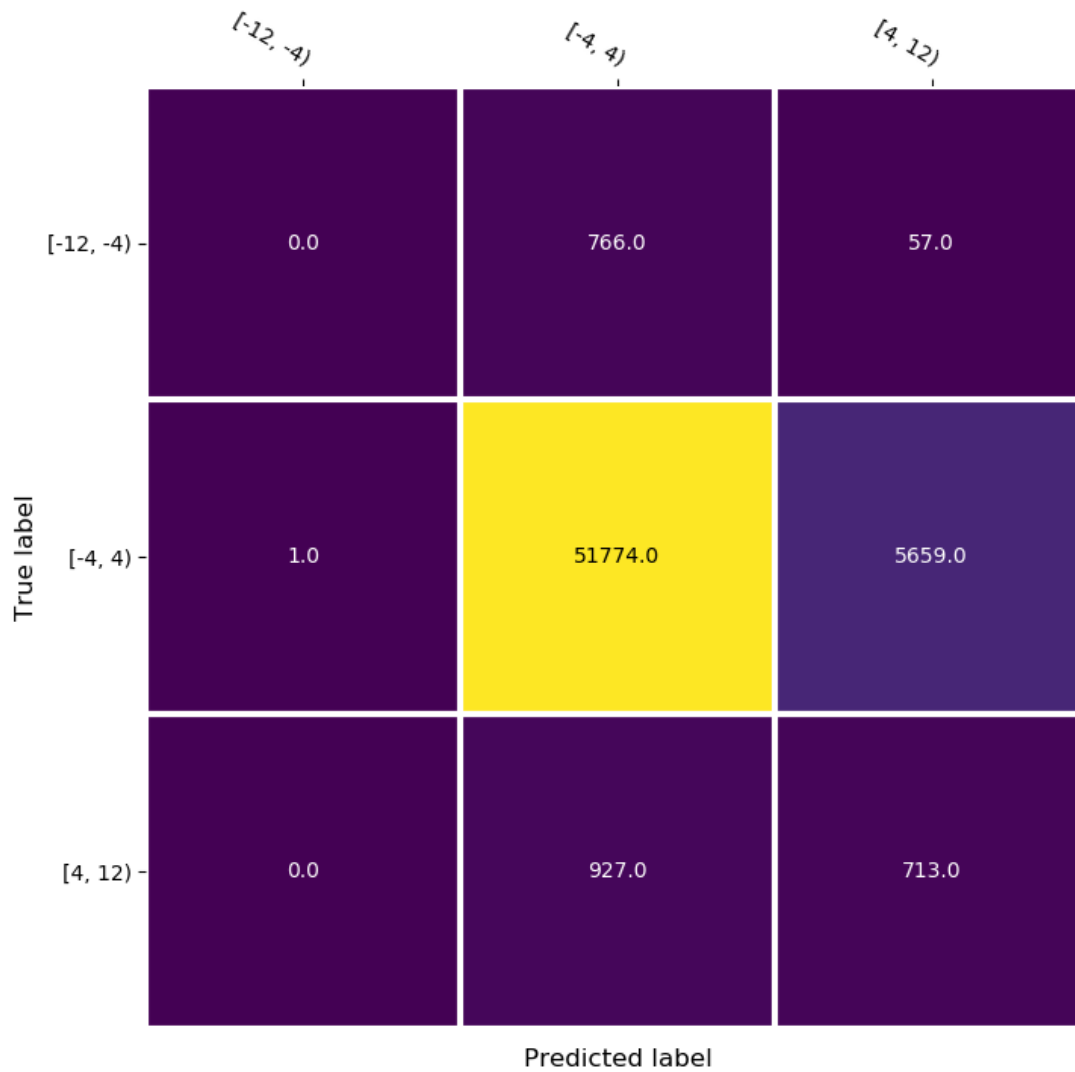
Classification Report:

	precision	recall	f1-score	support
[-12, -4)	0.00	0.00	0.00	823
[-4, 4)	0.97	0.90	0.93	57434
[4, 12)	0.11	0.43	0.18	1640
accuracy			0.88	59897
macro avg	0.36	0.45	0.37	59897
weighted avg	0.93	0.88	0.90	59897

Accuracy : 87.62876270931767

The Accuracy is 87.63, we improve our model.

Confusion Matrix:



We can see that target value $[-4,4)$ is the highest number of correct predictions.

5.5 KNN

- o Ranging by interval 1

KNN classification report

Classification Report:

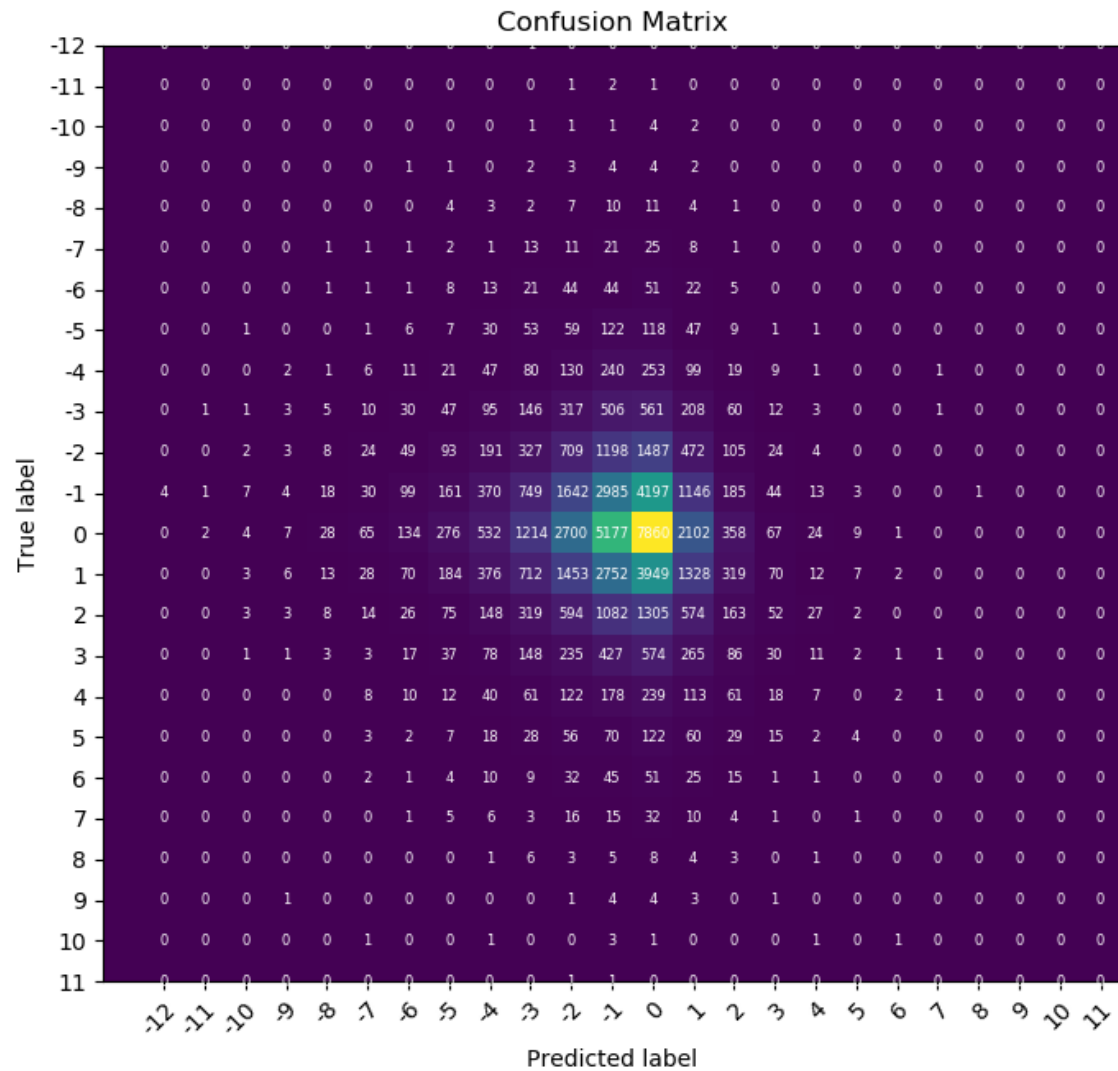
	precision	recall	f1-score	support
0	0.00	0.00	0.00	1
1	0.00	0.00	0.00	4
2	0.00	0.00	0.00	9
3	0.00	0.00	0.00	17
4	0.00	0.00	0.00	42
5	0.01	0.01	0.01	85
6	0.00	0.00	0.00	211
7	0.01	0.02	0.01	455
8	0.02	0.05	0.03	920
9	0.04	0.07	0.05	2006
10	0.09	0.15	0.11	4696
11	0.20	0.26	0.22	11659
12	0.38	0.38	0.38	20560
13	0.20	0.12	0.15	11284
14	0.11	0.04	0.06	4395
15	0.09	0.02	0.03	1920
16	0.06	0.01	0.01	872
17	0.14	0.01	0.02	416
18	0.00	0.00	0.00	196
19	0.00	0.00	0.00	94
20	0.00	0.00	0.00	31
21	0.00	0.00	0.00	14
22	0.00	0.00	0.00	8
23	0.00	0.00	0.00	2
accuracy			0.22	59897
macro avg	0.06	0.05	0.05	59897
weighted avg	0.23	0.22	0.22	59897

Accuracy : 22.184750488338313

5.843948778736832

Accuracy is 22.18 and MSE is 5.84

Confusion Matrix:



We can see that target value 0 is the highest number of correct predictions.

o Ranging by interval 8

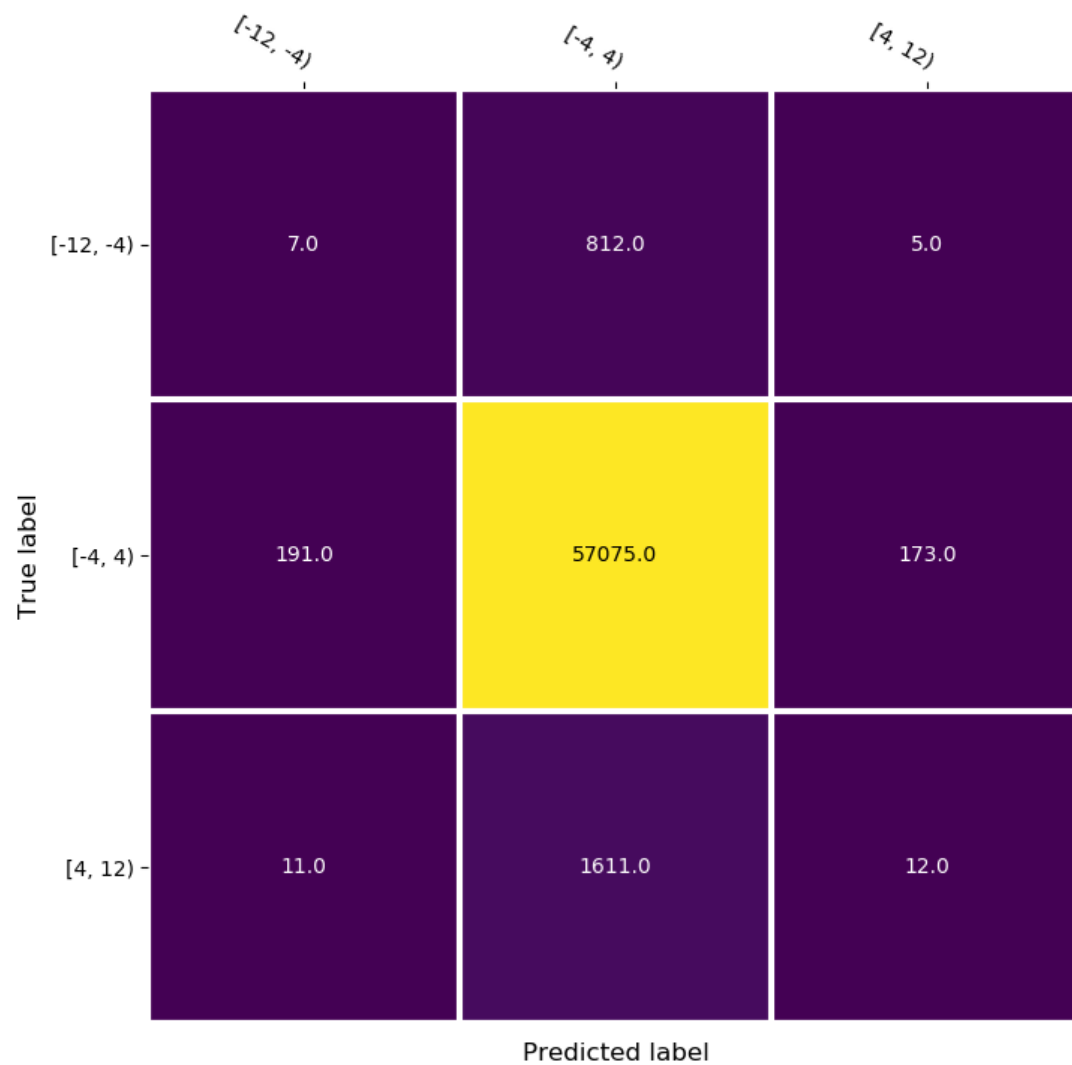
KNN classification report

Classification Report:

	precision	recall	f1-score	support
0	0.03	0.01	0.01	824
1	0.96	0.99	0.98	57439
2	0.06	0.01	0.01	1634
accuracy			0.95	59897
macro avg	0.35	0.34	0.33	59897
weighted avg	0.92	0.95	0.94	59897

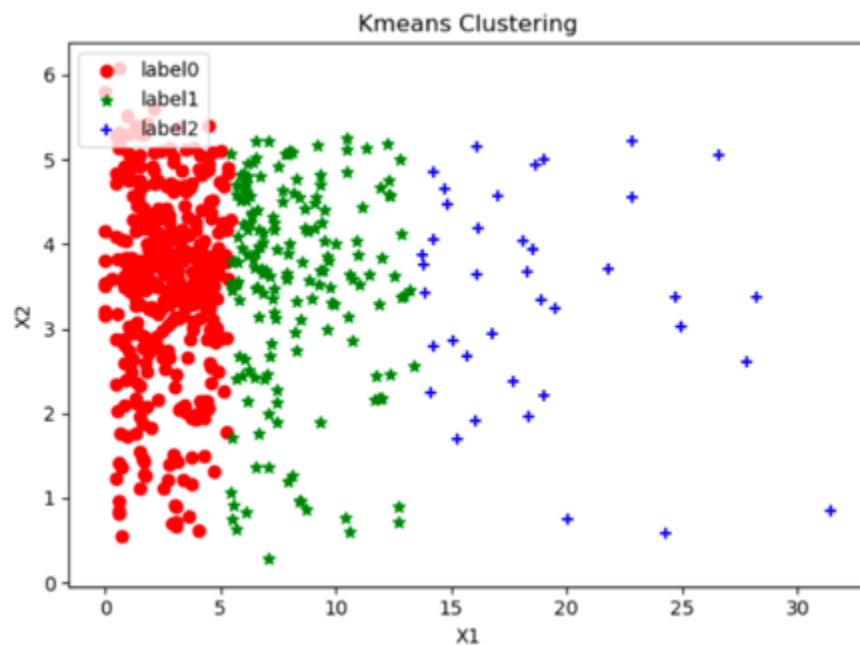
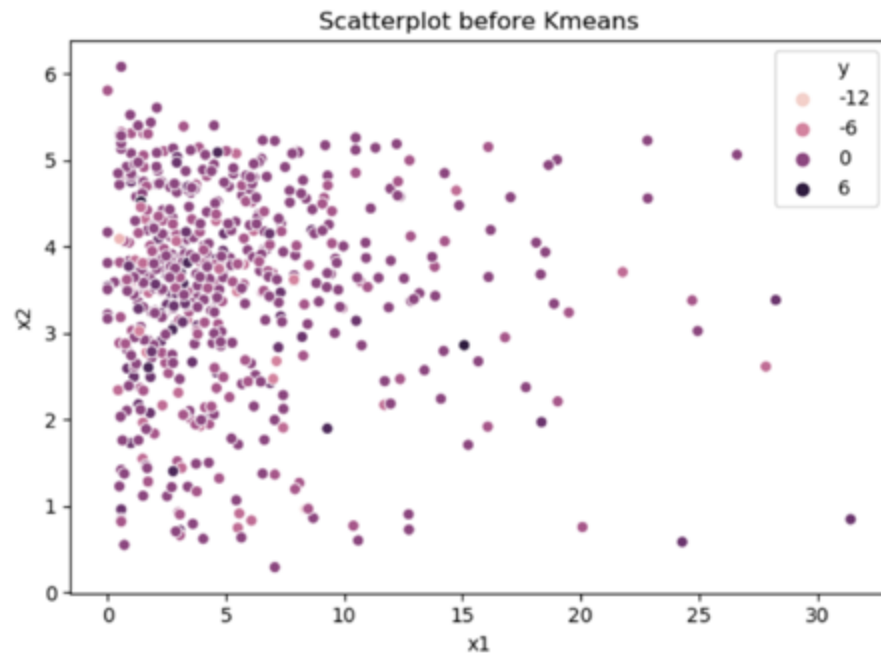
Accuracy : 95.32029984807252

The accuracy is 95.32%. It is very good
Confusion Matrix



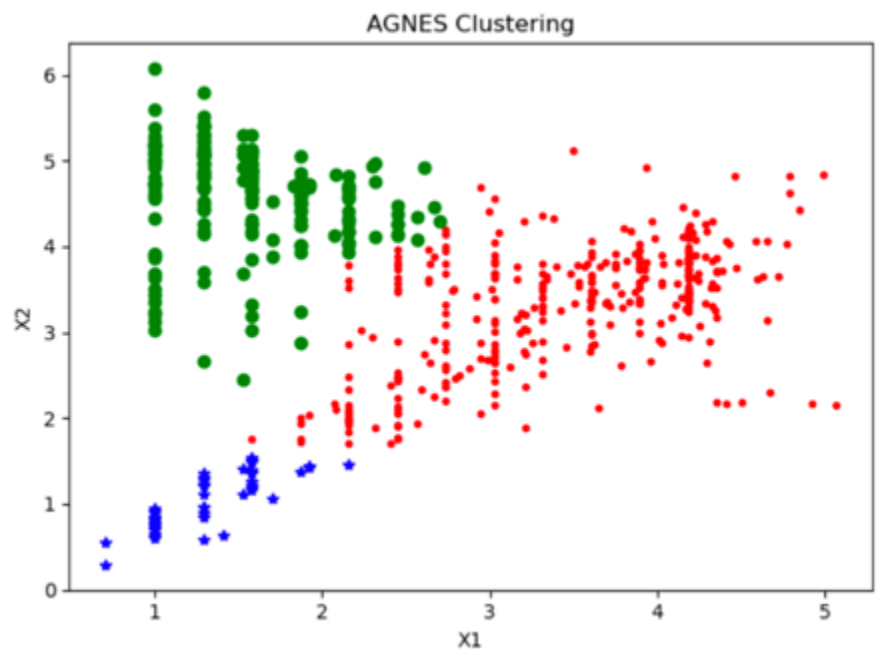
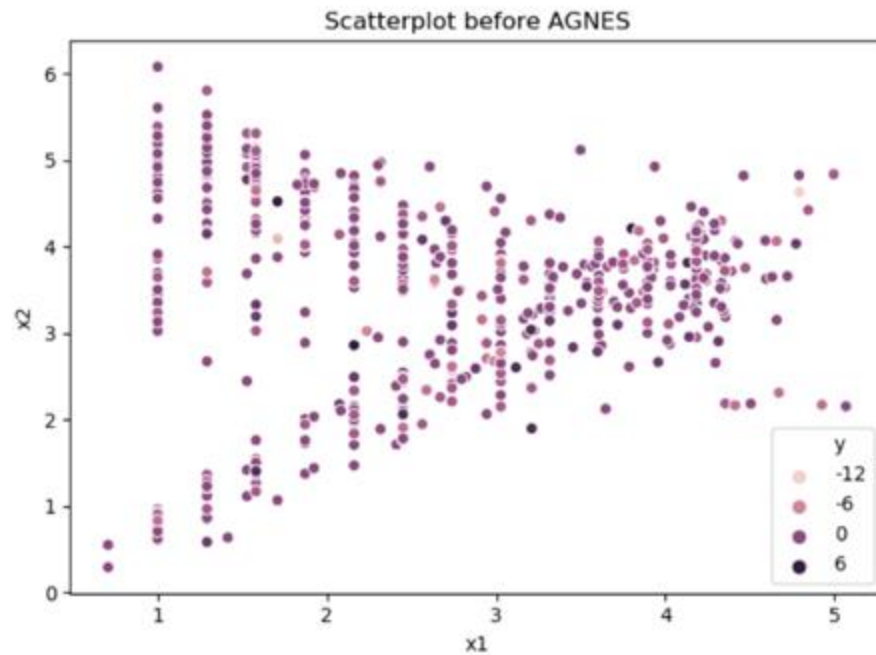
We can see that target value $[-4,4)$ is the highest number of correct predictions.

5.5.1 K-means



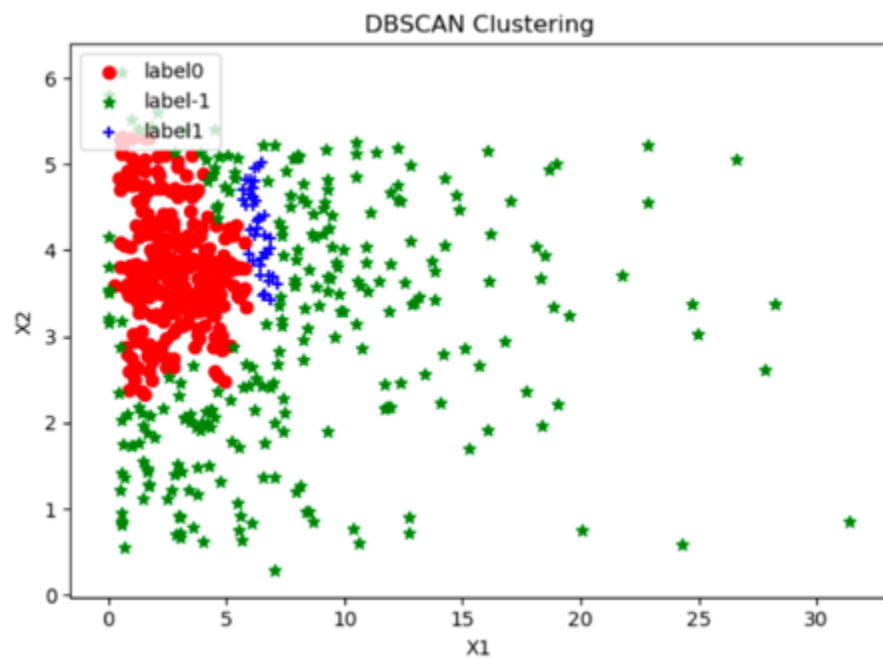
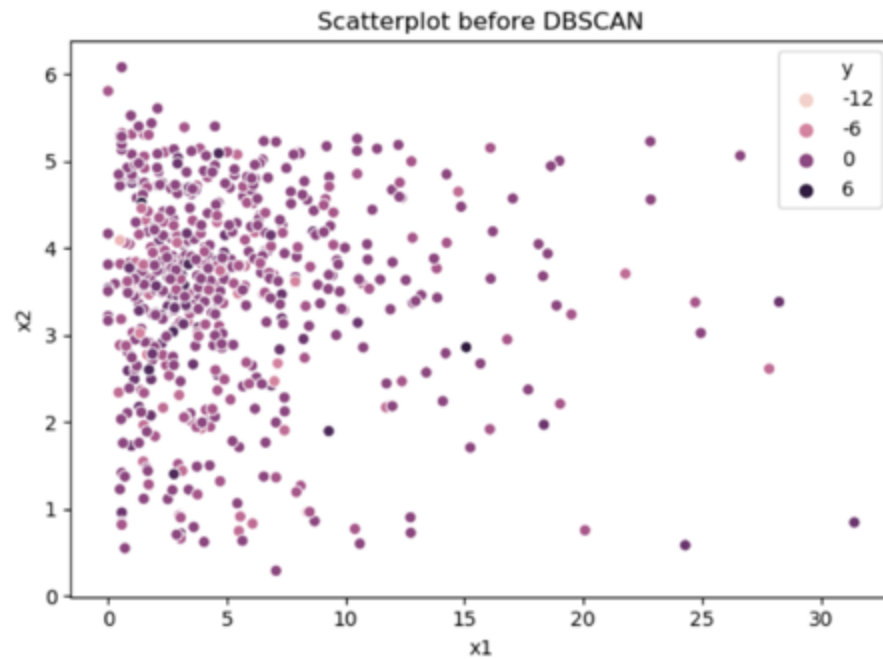
When features are selected as "purchase_amount_count_std" and "auth_purchase_month_std", scatter plots of raw data and result data after K-Means Clustering.

5.6 AGNES



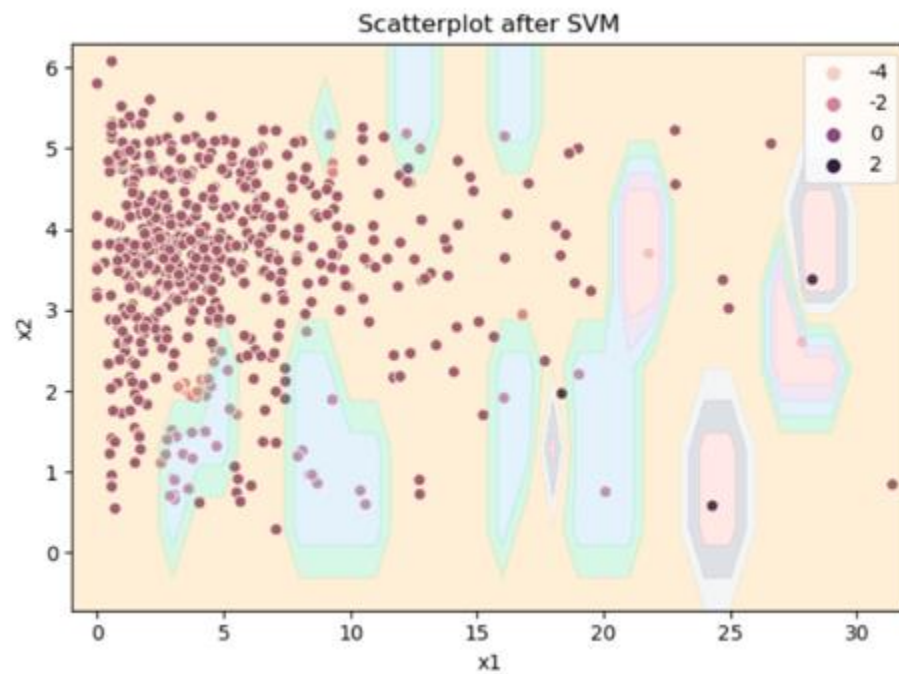
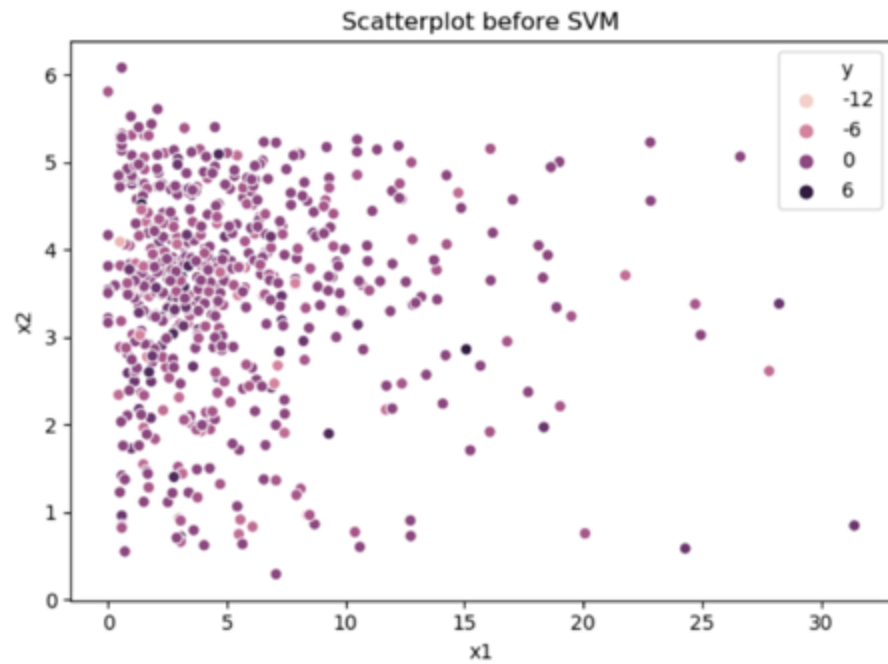
When features are selected as "month_lag_std" and "auth_purchase_month_std", scatter plots of raw data and result data after AGNES.

5.7 DBSCAN

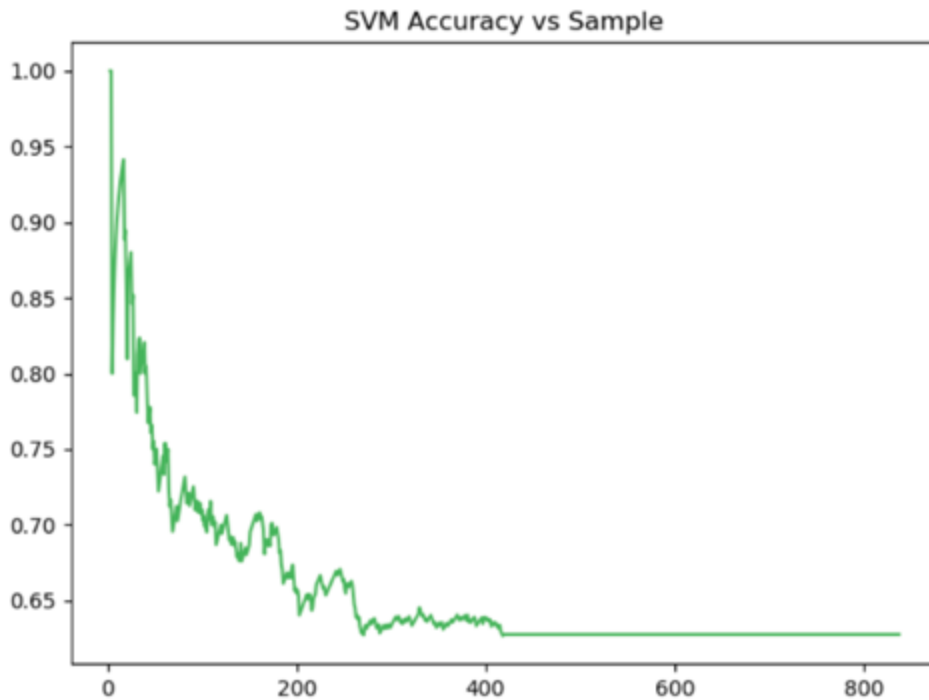


When features are selected as "purchase_amount_count_std" and "auth_purchase_month_std", scatter plots of raw data and result data after DBSCAN.

5.8 SVM



When features are selected as "purchase_amount_count_std" and "auth_purchase_month_std", scatter plots of raw data and result data after SVM.



The chart shows the relationship between sample size and accuracy in Support Vector Machine (SVM). It can be seen from the curve that as the sample size increases, the accuracy of the SVM gradually stabilizes, and finally remains at about 62.5%.

6. Summary and conclusions. Summarize the results you obtained, explain what you have learned, and suggest improvements that could be made in the future.

For continuous target, linear regression is the one model to be used for prediction. However, decision tree fitted using converted categorical target performs well, as the MSE is 2.87.

As for improvements that could be done for this project, we could try LGBost model which uses a novel technique of Gradient-based One-Side Sampling (GOSS) to filter out the data instances for finding a split value.

The accuracy for interval of 1 is very low for all four models (Decision Tree, Random Forest, KNN, Naïve Bayes). But MSE it is ok for decision tree, random forest and naive bayes except KNN. KNN is not a good model to deal with lots of features dataset. I believe that the reason for this problem is that the raw data (the target column `y_train`) is a continuous variable, so we changed this column from continuous to category. For ranging by interval 1, we have 24 categories; it will decrease the accuracy. The random forest and decision tree have higher accuracy, it is more fit than other model for this dataset. The accuracy for interval of 4 and 8 was improved, suggesting that less categories will influence the accuracy. Our model is quite ok. In ranging interval of 8, we

still have 3 categories, each model has high accuracy. Which means improve our model successful. Our model is a good model.

I think the confusion matrix has some problems (bug) about the x and y axis and I used a class example for reference. It has the same problem. The number of corrections in the confusion matrix is correct.

When the variables are selected the same, the results of the two methods of K-means and AGNES are roughly the same, but the results of DBSCAN and the former are somewhat different.

K-means is a simple and efficient way for large data sets, with low time and space complexity. The most important thing is that the results are easy to be locally optimal when the data set is large; K values need to be set in advance, which is sensitive to the selection of the first K points; very sensitive to noise and outliers; only used for numerical data; cannot solve non- Convex data.

DBSCAN is not sensitive to noise and it can find clusters of any shape. However, the results of clustering have a great relationship with parameters. DBSCAN uses fixed parameters to identify clusters, but when the degree of sparseness of the clusters is different, the same judgment criteria may destroy the natural structure of the clusters, that is, the thinner clusters Will be divided into multiple classes or denser and closer classes will be merged into a cluster.

It can be seen from the graph of the SVM test that as the number of samples continues to increase, the accuracy of the model will gradually decrease, and will stabilize when the number of samples reaches a certain number.

Since SVM uses the quadratic programming to solve the support vector, solving the quadratic programming will involve the calculation of a matrix of order m (m is the number of samples). When the number of m is large, the storage and calculation of the matrix will consume a lot of machines. Memory and operation time. In view of the above problems, in the future research, the following methods can be used to improve the model: J.Platt's SMO algorithm, T. Joachims 'SVM, C.J.C.Burges' PCGC, Zhang Xuegong's CSVM, and O.L. Mangasarian's SOR algorithm.

7. References

<https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba>

Elo Merchant Category Recommendation Help understand customer loyalty. Elo. (March, 2019). Retrieved from <https://www.kaggle.com/c/elo-merchant-category-recommendation>

8. A separate appendix should contain documented computer listings.

<https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.fillna.html>