Elo Merchant Category Recommendation

-Help understand customer loyalty

6103 Group 4

**1. Introduction**

[Elo](https://www.cartaoelo.com.br/), 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]](http://github.com/pandas-dev/pandas/blob/v0.25.3/pandas/core/frame.py#L4240-L4260)

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 integrals ranging from -12 to 12 into

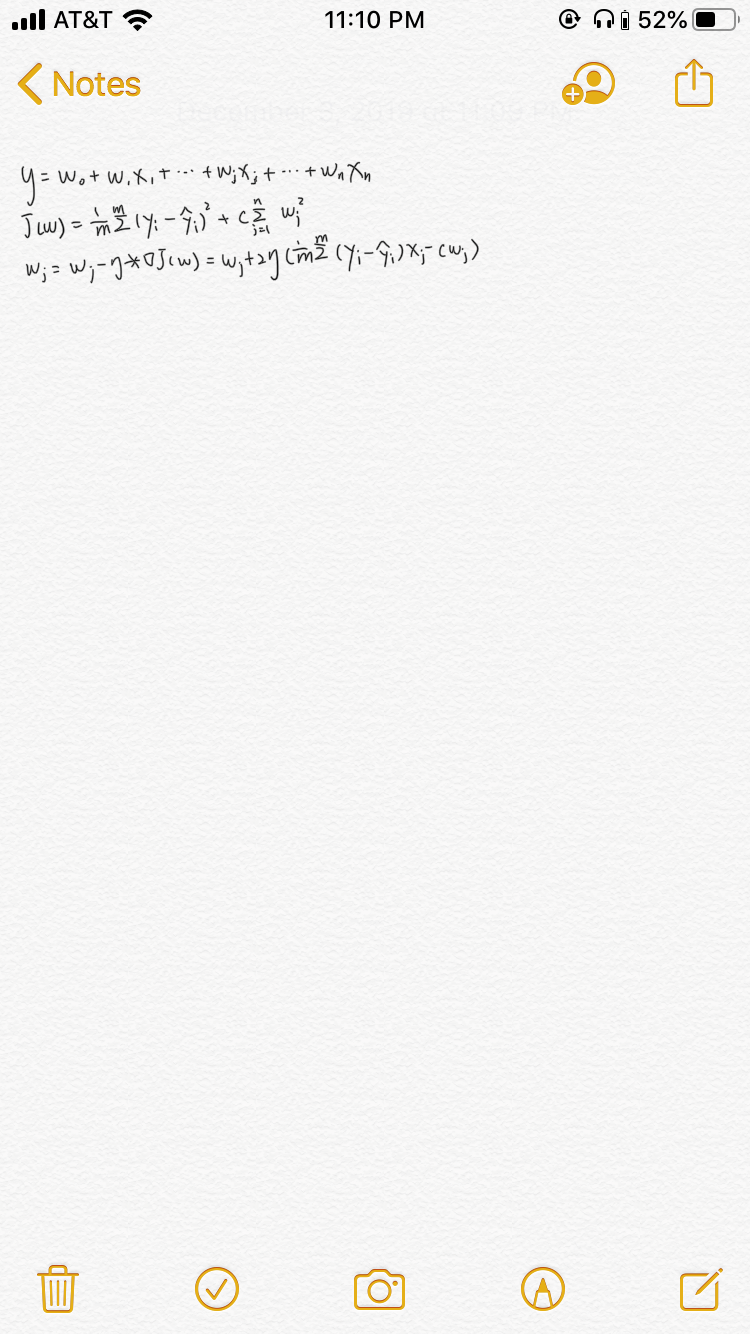
· 24 groups with an interval of 1

· 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)



Assumption: y = β0 + β1x + err

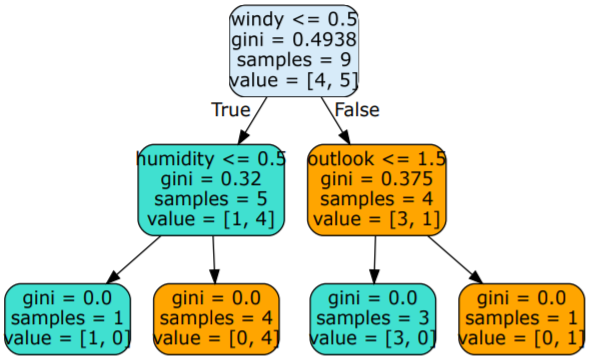
Goal: estimate β0 and β1 based on the available data

Final Model ˆy = βˆ 0 + βˆ 1x + err

β0 and β1 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](https://en.wikipedia.org/wiki/Ensemble_learning) method for [classification](https://en.wikipedia.org/wiki/Statistical_classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis) and other tasks that operates by constructing a multitude of [decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning) at training time and outputting the class that is the [mode](https://en.wikipedia.org/wiki/Mode_(statistics)) of the classes (classification) or mean prediction (regression) of the individual tree. Random decision forests correct for decision trees' habit of [overfitting](https://en.wikipedia.org/wiki/Overfitting) to their [training set](https://en.wikipedia.org/wiki/Test_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

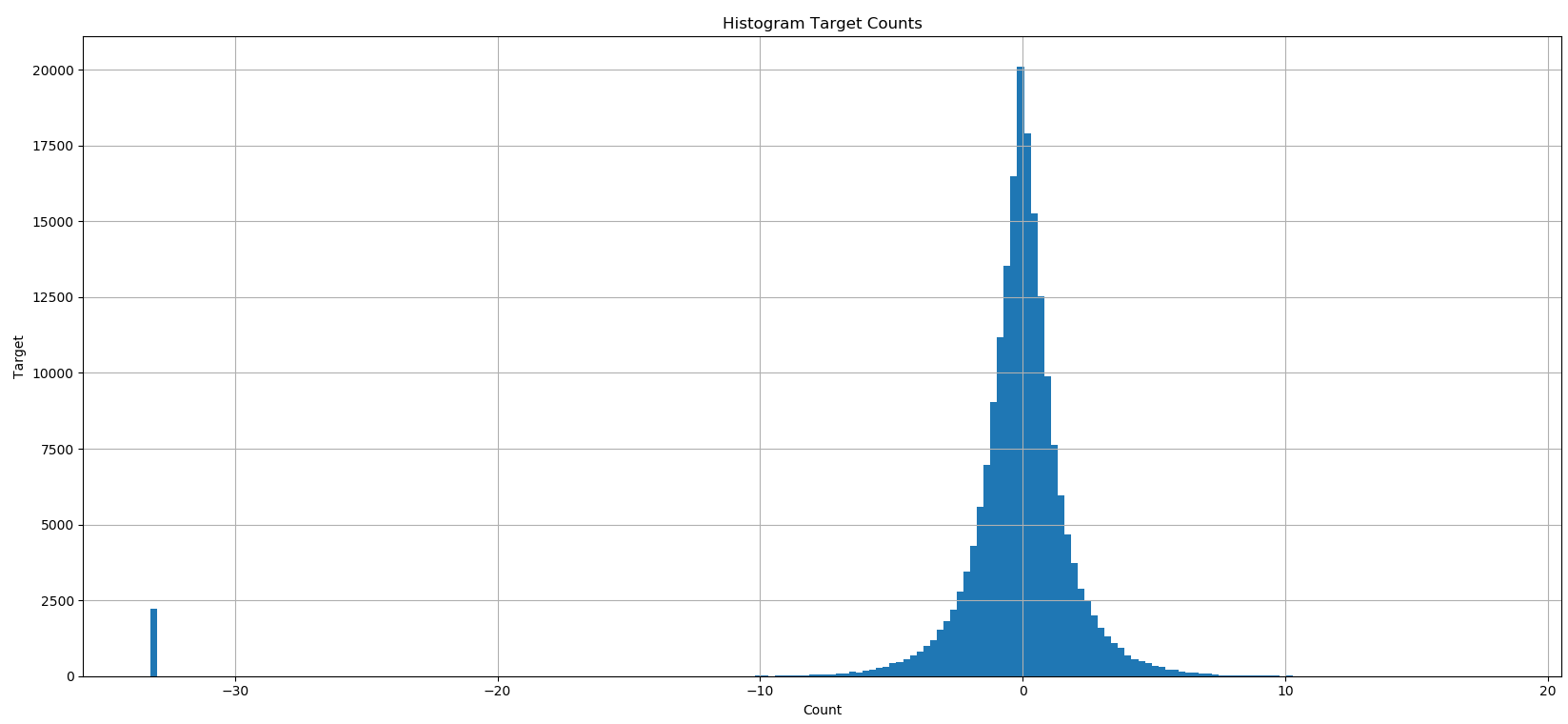
Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a very popular density based data clustering algorithm commonly used in data mining and machine learning. DBSCAN clusters the data points to separate the areas of high density with areas of low density. It also marks data points 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.**

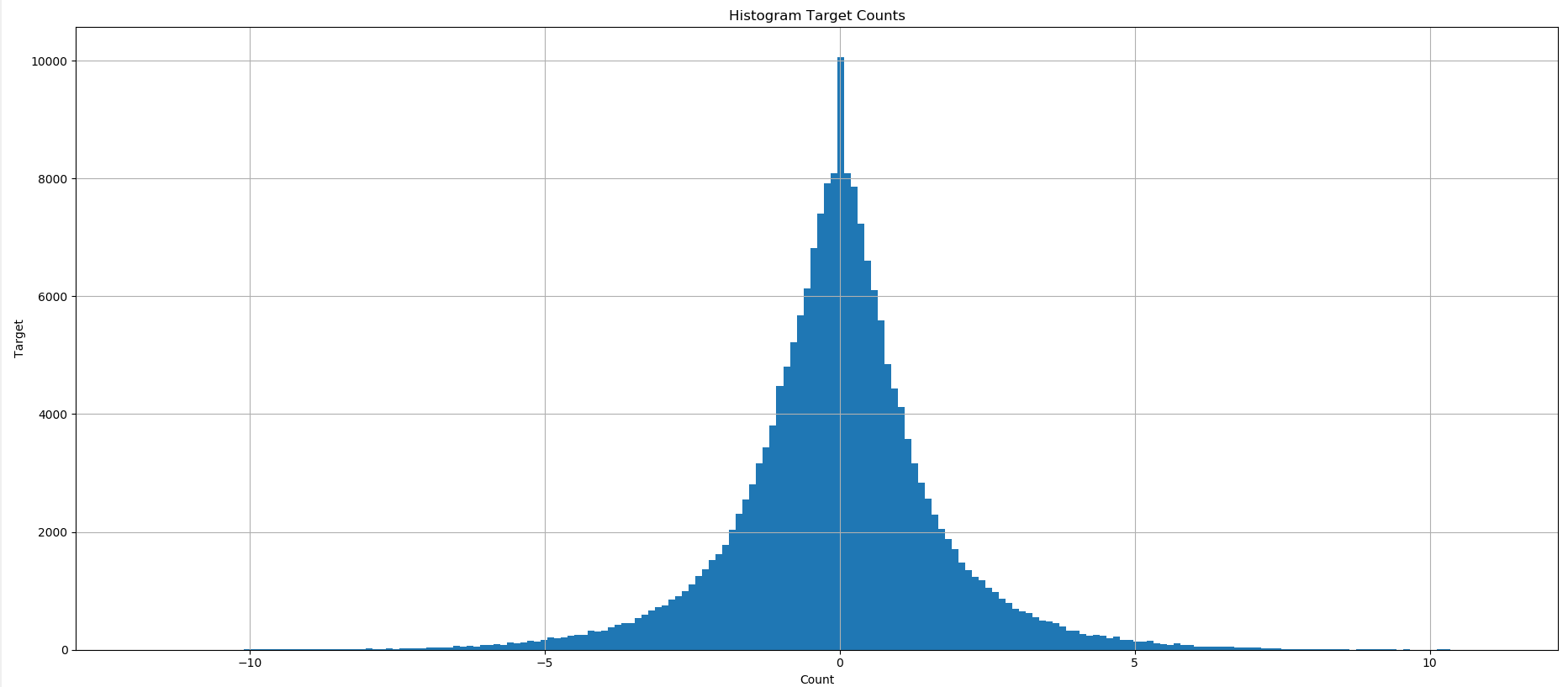
4.1 Preprocessing and feature engineering

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



1. Get dummies of 'feature\_1', 'feature\_2' and 'feature\_3' in historical\_transactions and new\_merchant\_transactions, train and test.
2. 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.
3. Fill missing values after feature engineering with 0
4. 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)

1. Using LabelEncoder() to transfer the target.
2. Using StandardScaler() to standardize X\_train and X\_test
3. Using train\_test\_split from sklearn.model\_selection, divide the data into training and testing (with test\_size=0.3 and random\_state = 0)
4. Use GridSearchCV from sklearn.model\_selection to tune hyperparameters such as max\_depth, min\_samples\_leaf and min\_samples\_split
5. Train a decision tree with criterion as gini and best hyperparameter selected using X\_train and y\_train.
6. Make predictions y\_pred using the trained model and X\_test
7. Calculate accuracy and mean\_square\_error using y\_pred and y\_test.
8. Get confusion matrix
9. Display decision tree.

4.2.3 Random forest

1. Import dataset
2. Split dataset
   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)

* 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, -4)', '[-4, 4)', '[4, 12)']

data['new\_target'] = pd.cut(data['target'], bins, labels=names)

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

4.2.4 Naive Bayes

1. Import packages
2. Split the dataset
   1. Split dataset
      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)

* + 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, -4)'**, **'[-4, 4)'**, **'[4, 12)'**]

1. Perform training
2. Make predictions
3. Calculate metrics
4. Confusion matrix

4.2.5 KNN

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

* + 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, -4)'**, **'[-4, 4)'**, **'[4, 12)'**]

1. Standardize the data
2. Perform training
3. Make predictions
4. Calculate metrics
5. Plot confusion matrix

**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.**

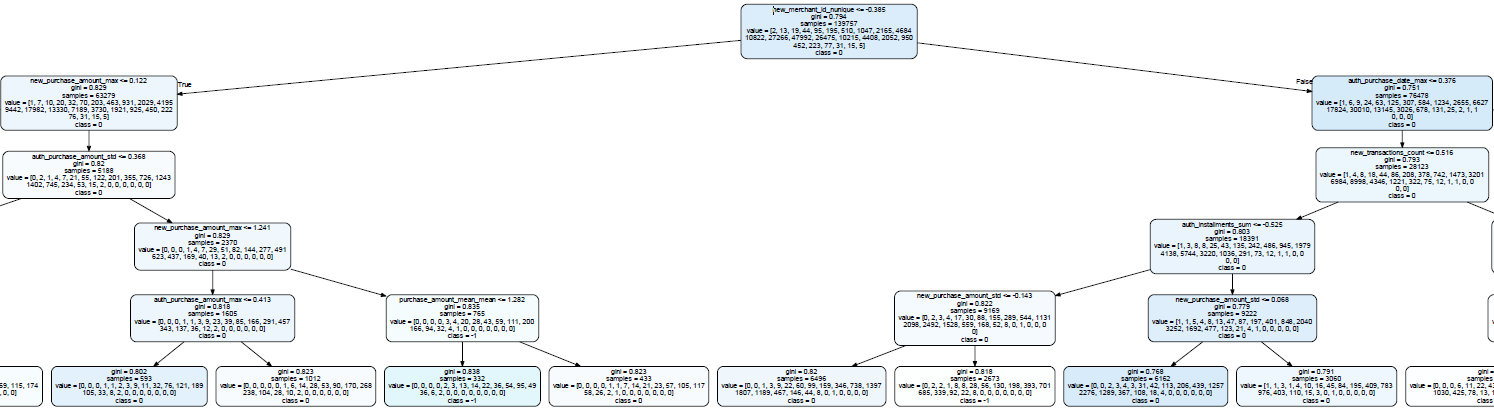
5.1 Linear Regression

Best\_score (*neg\_mean\_squared\_error)*: -0.8982704791215965; best\_params: {'estimator\_\_eta': 0.1}; Mean\_squared\_error: 2.65; r2\_score: 0.09

5.2 Decision Tree

Best hyperparameters are 'max\_depth': 6, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2

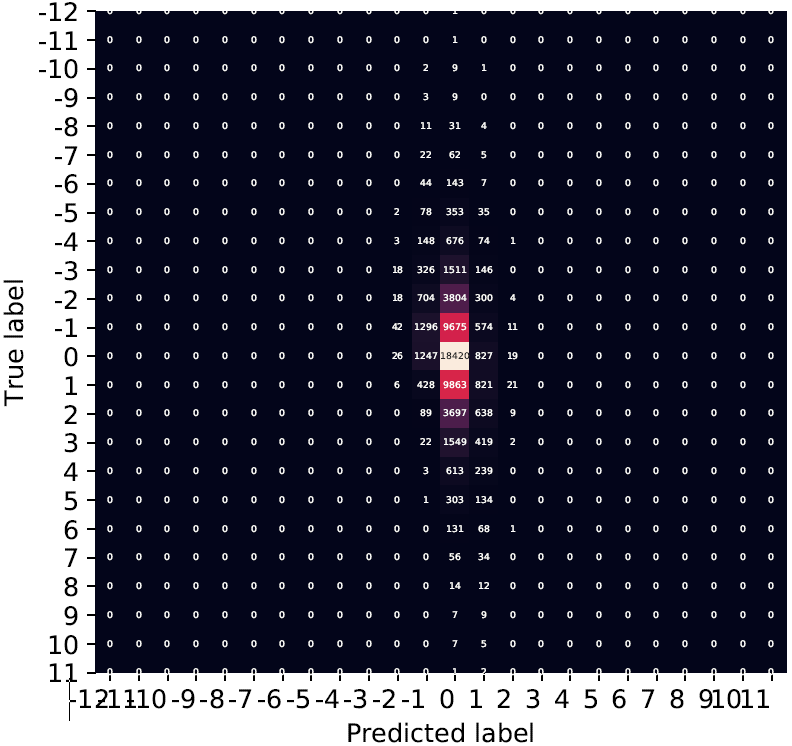
Part of the decision tree:



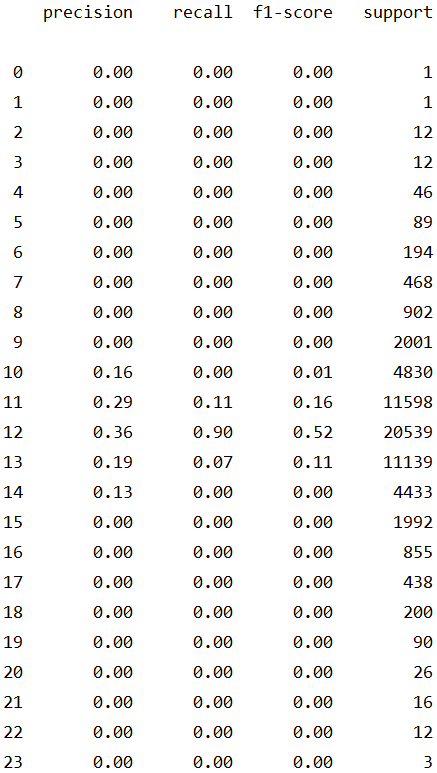
The most important feature selected by Decision Tree is new\_merchant\_id\_nunique <= 0

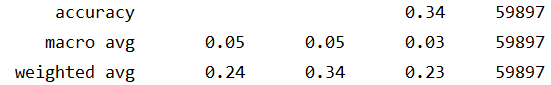
Accuracy: 34.33 and Mean\_square\_error: 2.87

Confusion matrix:



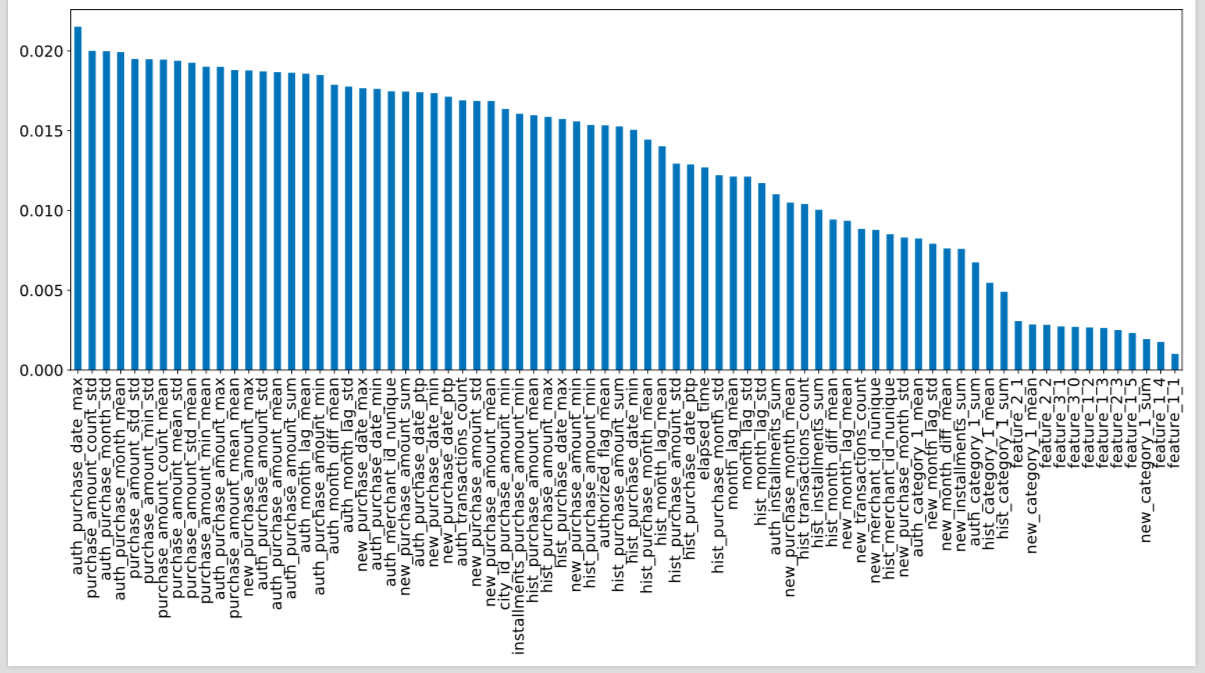
Classification Report:





5.3 Random Forest

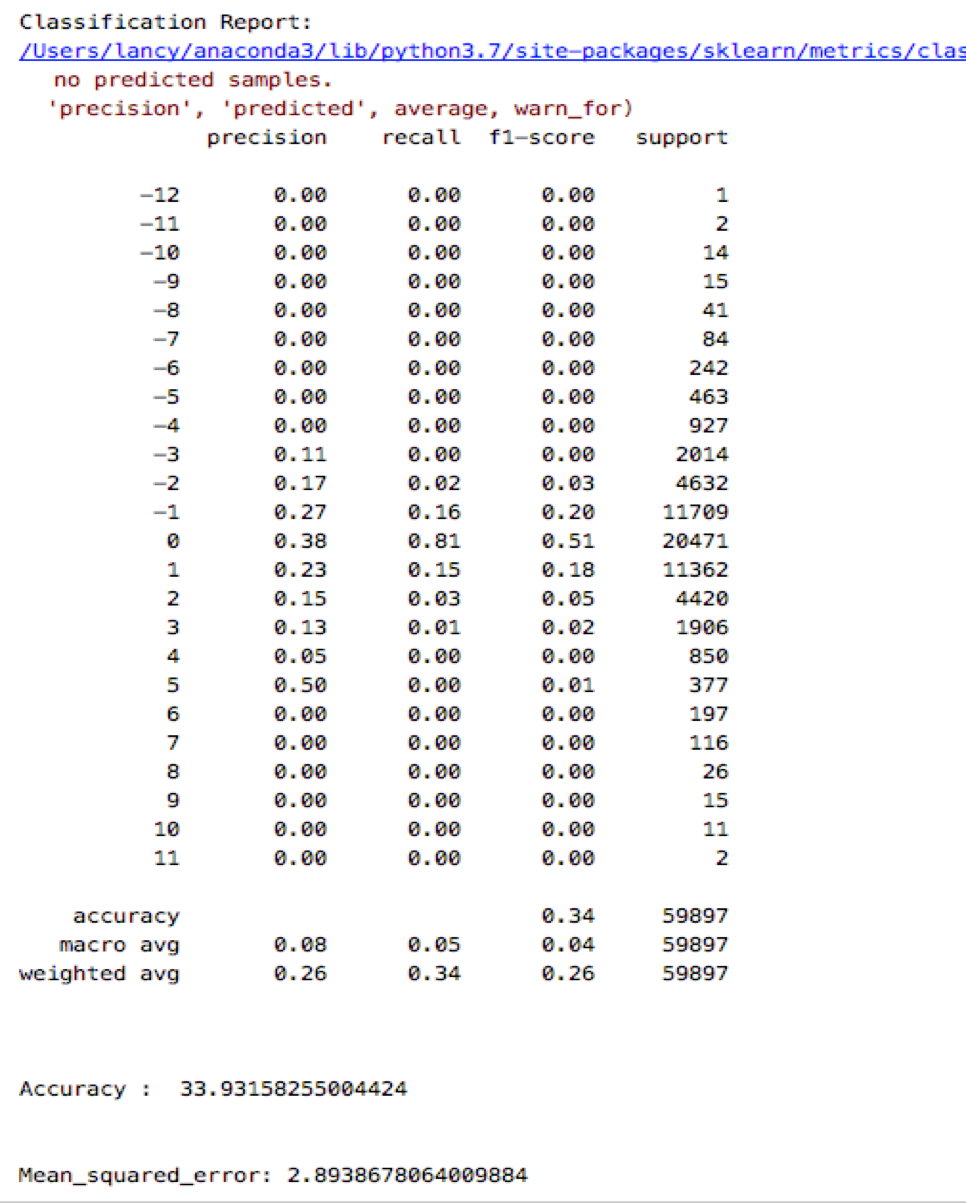
Features importance



We can see which feature is more important in this dataset, then we pick top 25 features to run the model.

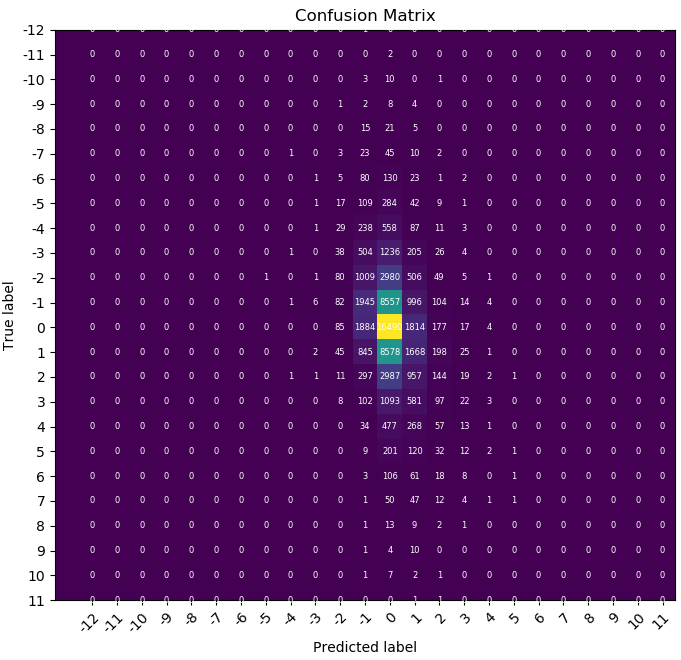
Ranging by interval 1

**Classification of All features**



Accuracy is 33.93 and MSE is 2.89

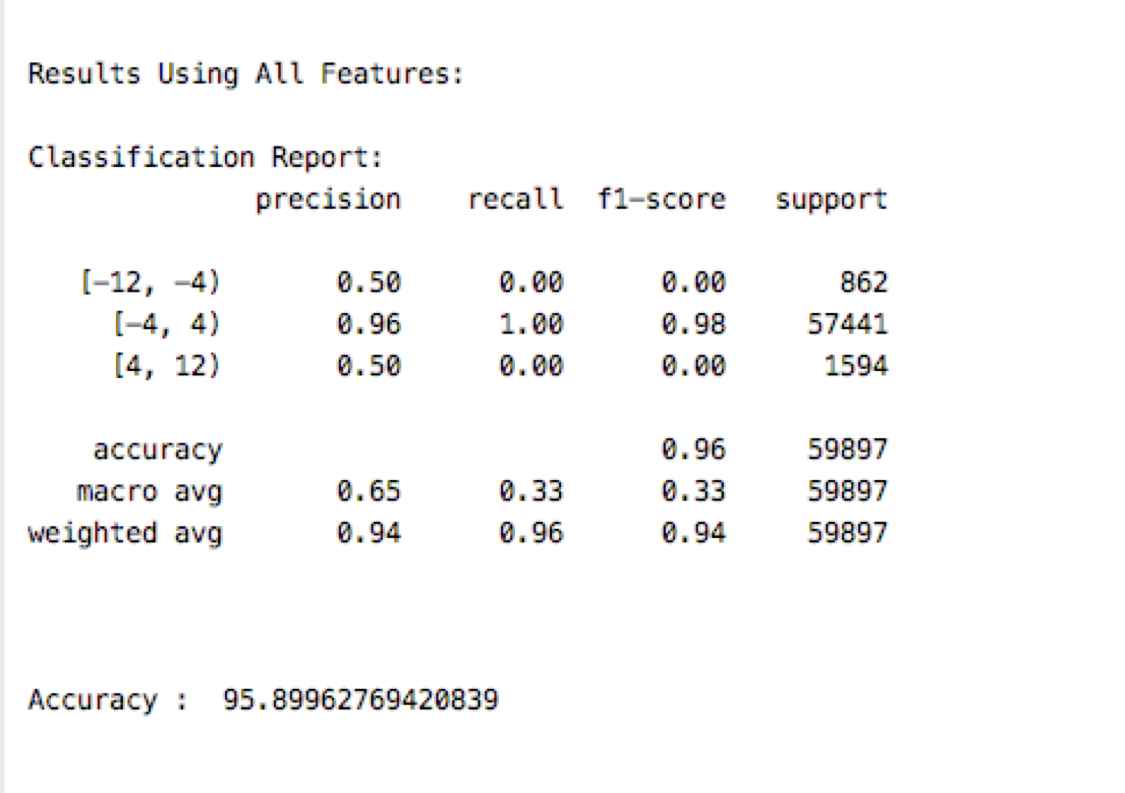
Confusion Matrix:



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

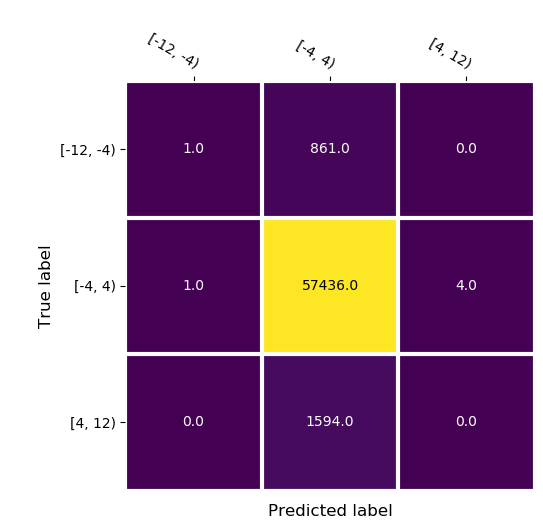
Ranging by interval 8

RF all features classification report



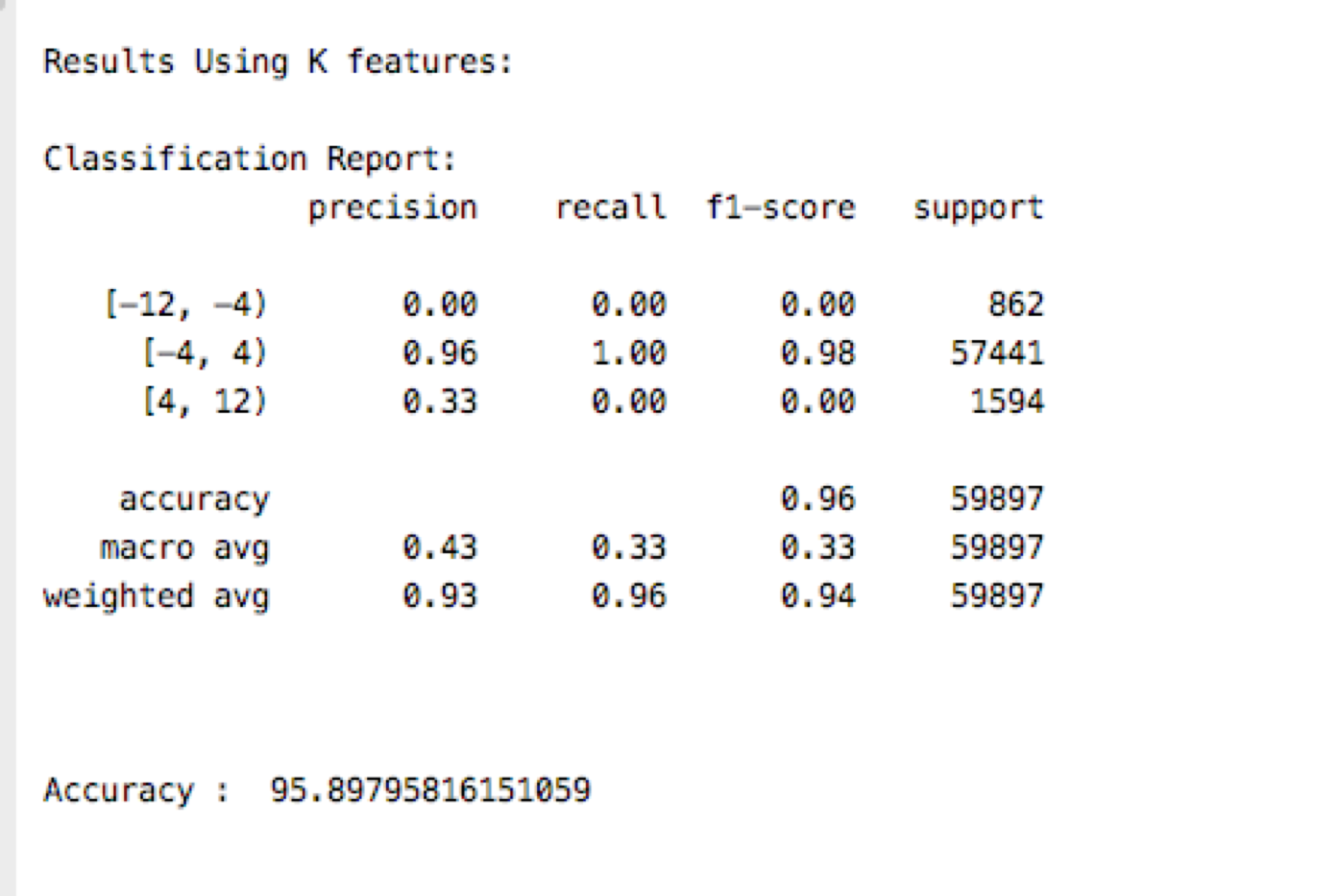
Accuracy is 95.89.

Confusion Matrix



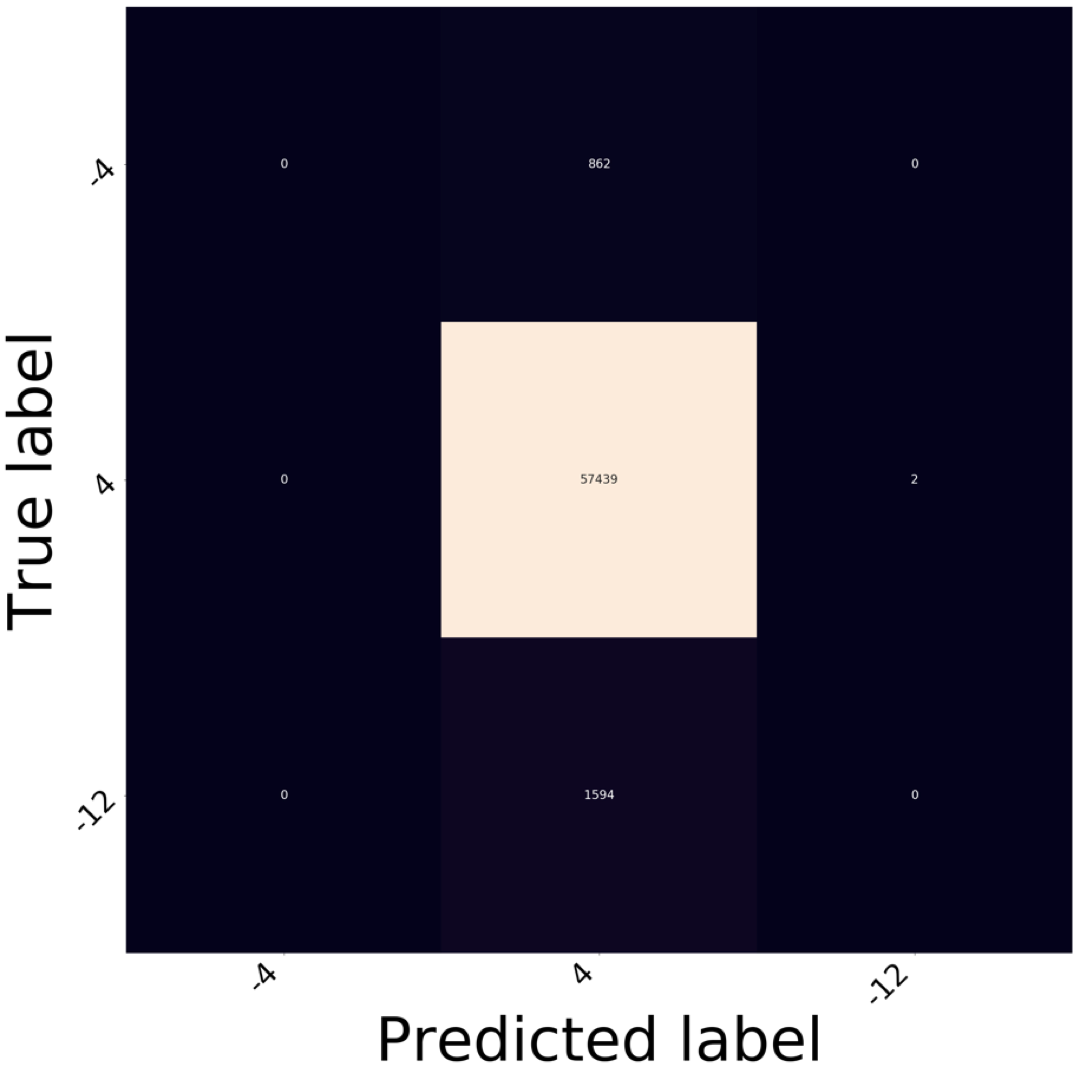
We can see that target value between [-4,4) is the highest number of correct predictions.

RF K features classification report



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:

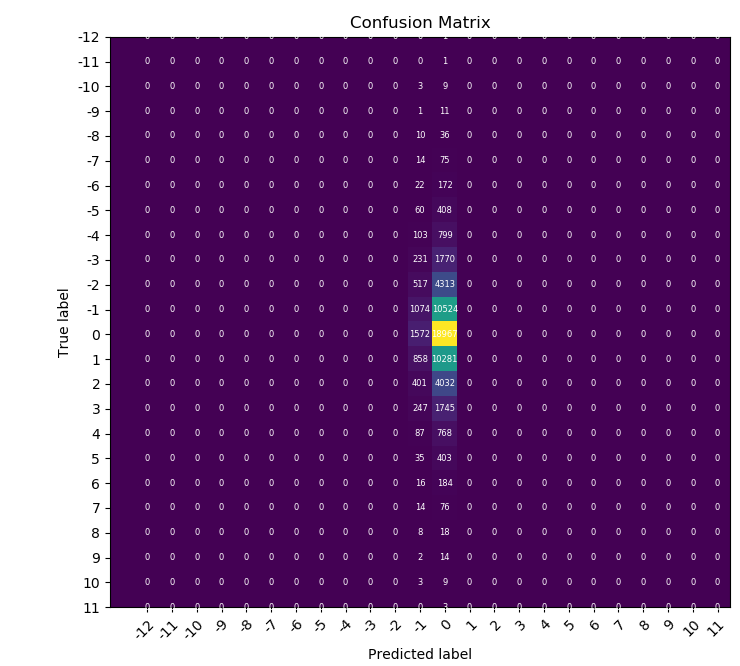
Fillna: Fill NA/NaN values using the specified method.

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.

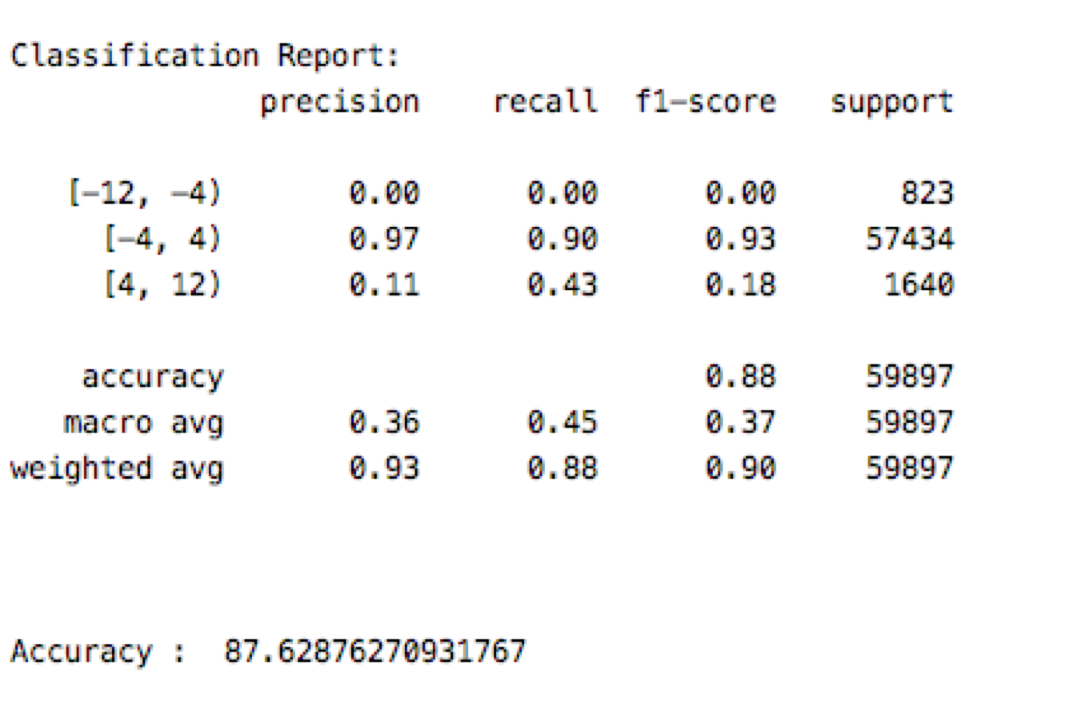
Confusion Matrix



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

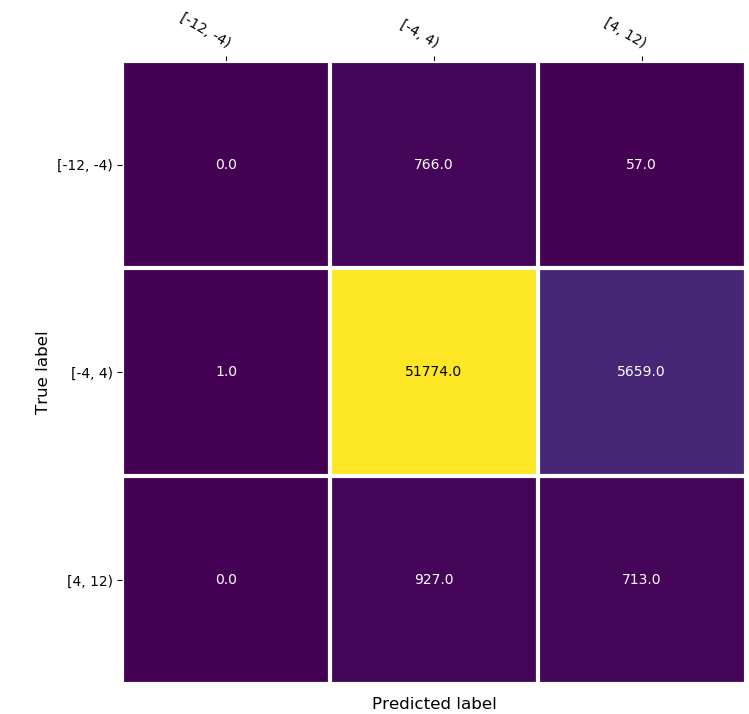
o Ranging by interval 8

Naive Bayes classification report



The Accuracy is 87.63.

Confusion Matrix:

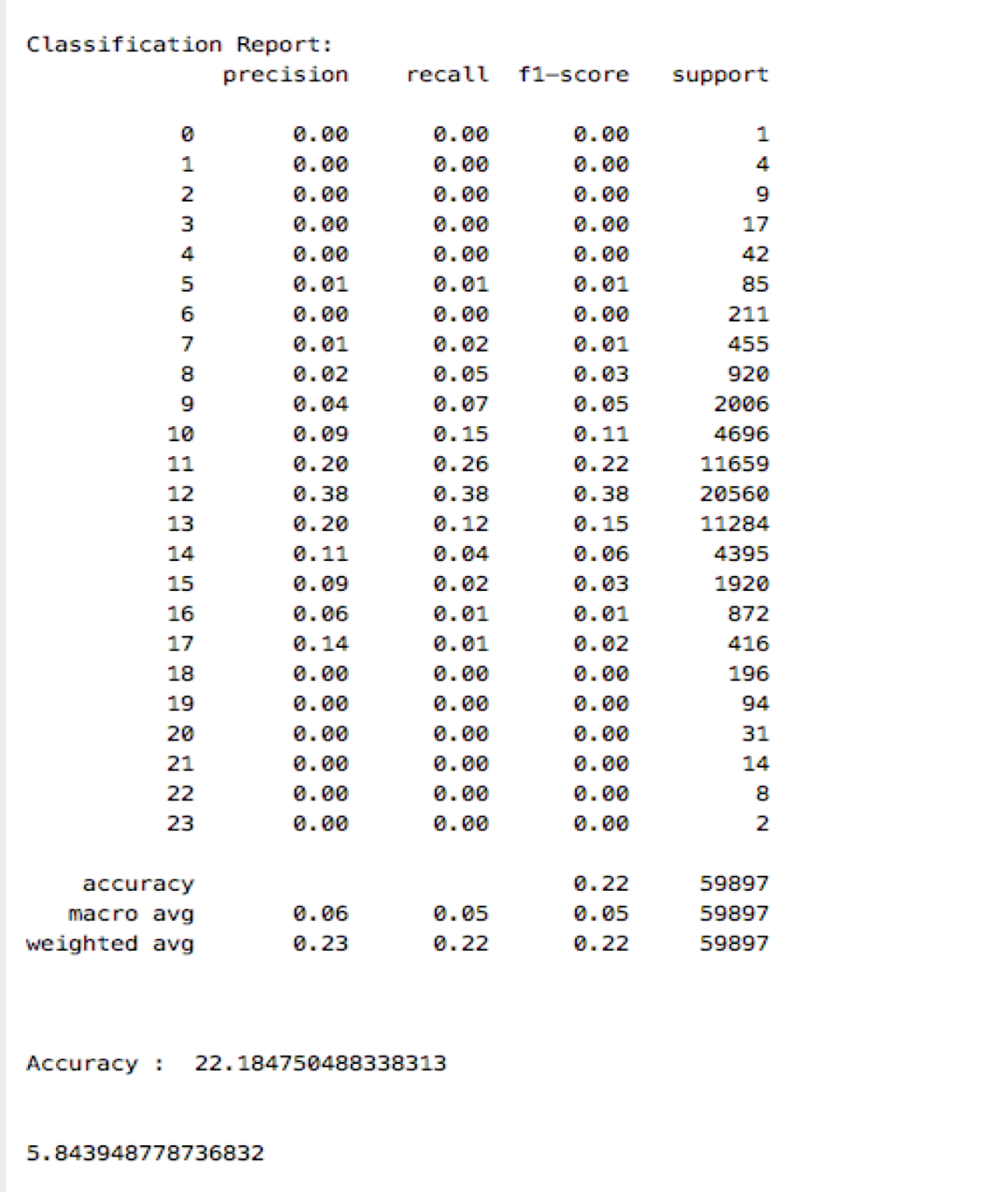


We can see that target value [-4,4) is the highest number of correct predictions.

5.5 KNN

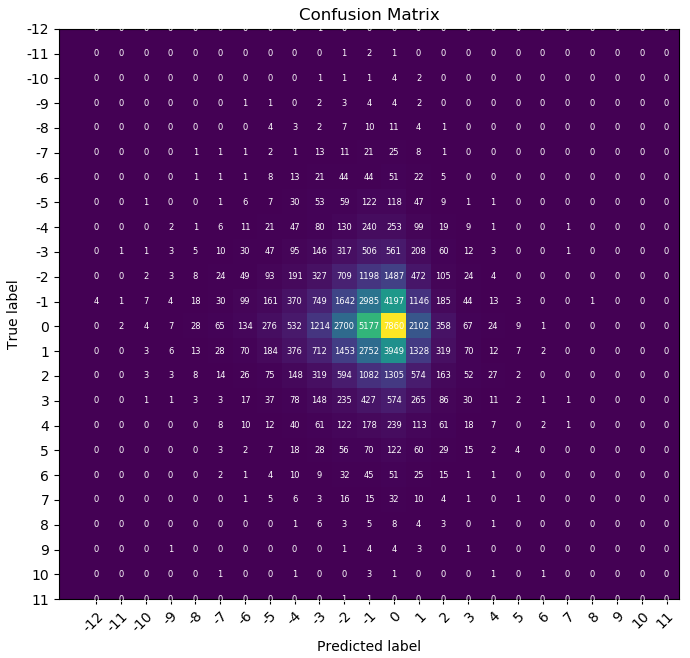
o Ranging by interval 1

KNN classification report



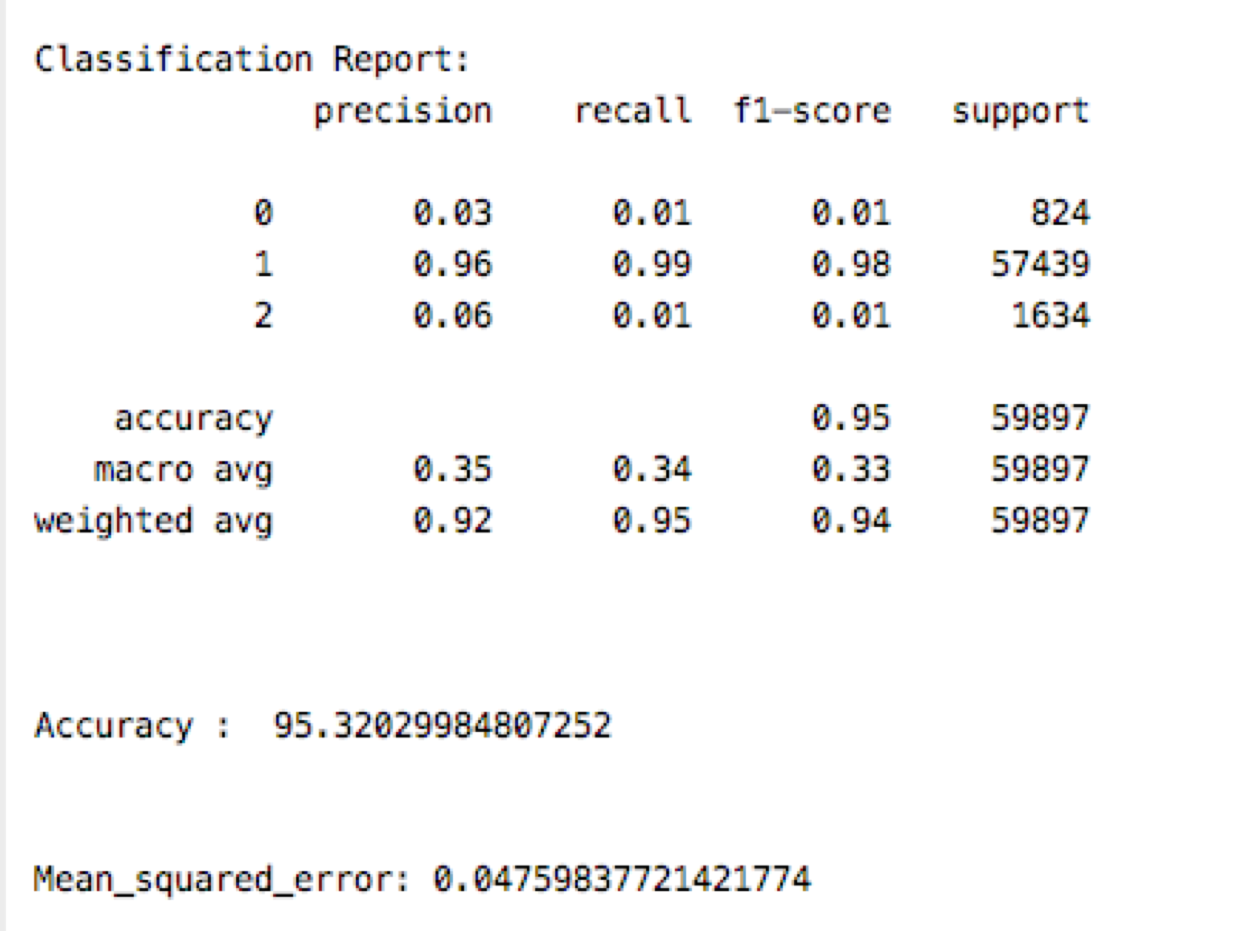
Accuracy is 22.18 and MSE is 5.84

Confusion Matrix:



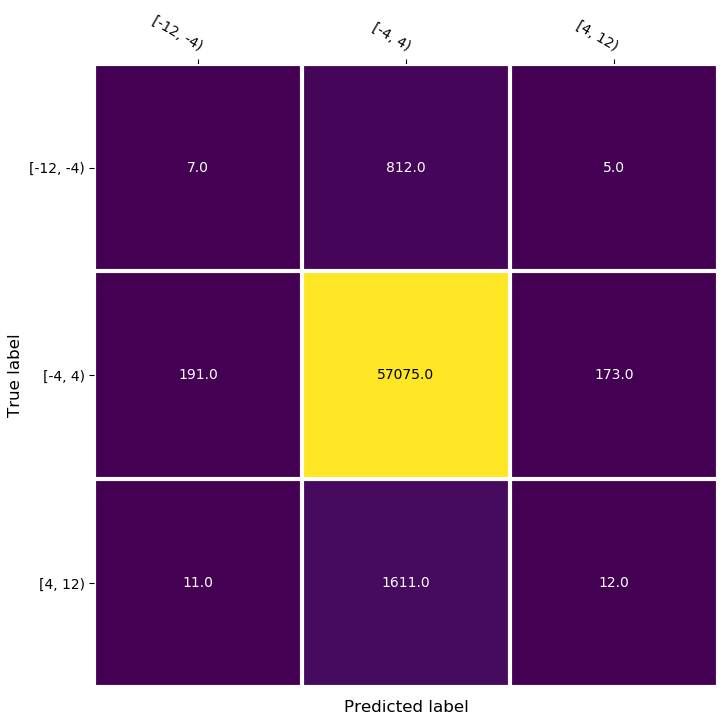
We can see that target value 0 is the highest number of correct predictions.

o Ranging by interval 8



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.

**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 LGBoost 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.

**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>