SP Capstone Project 1 Final Report



## **1. Define the Problem**

# Predict whether the customer will purchase again or not in the next 12 months after the first purchase is completed.

## **2. Prepare Data for Consumption**

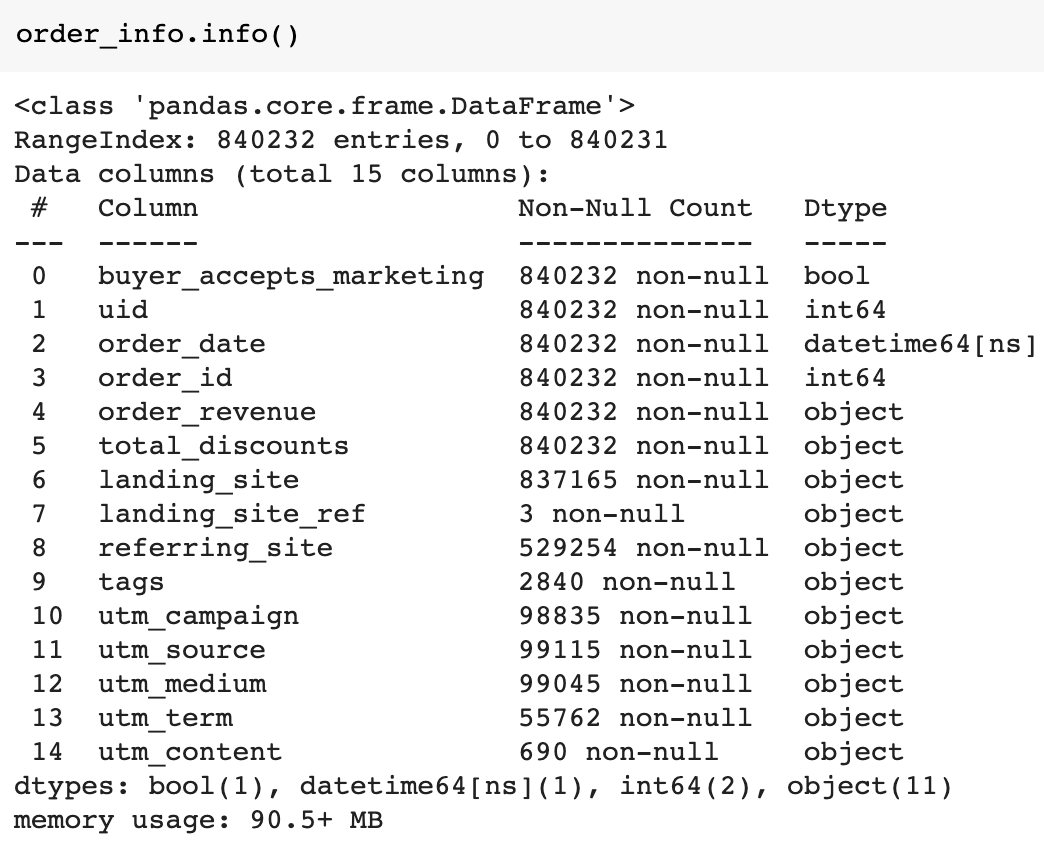
## **2.1 Gather the Data:**

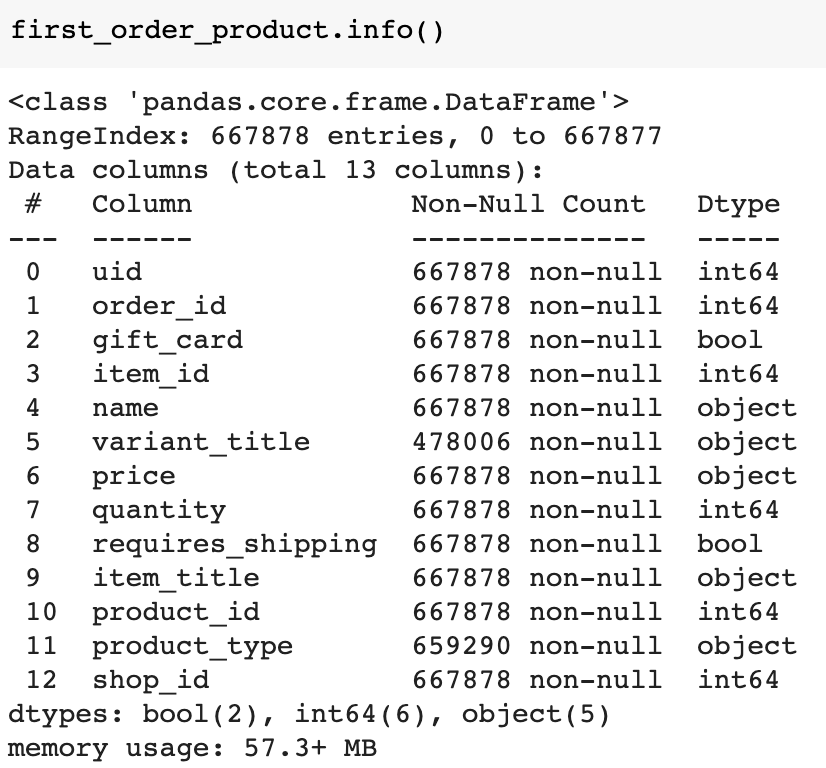
(1) Explore a ecommerce store data in multiple tables

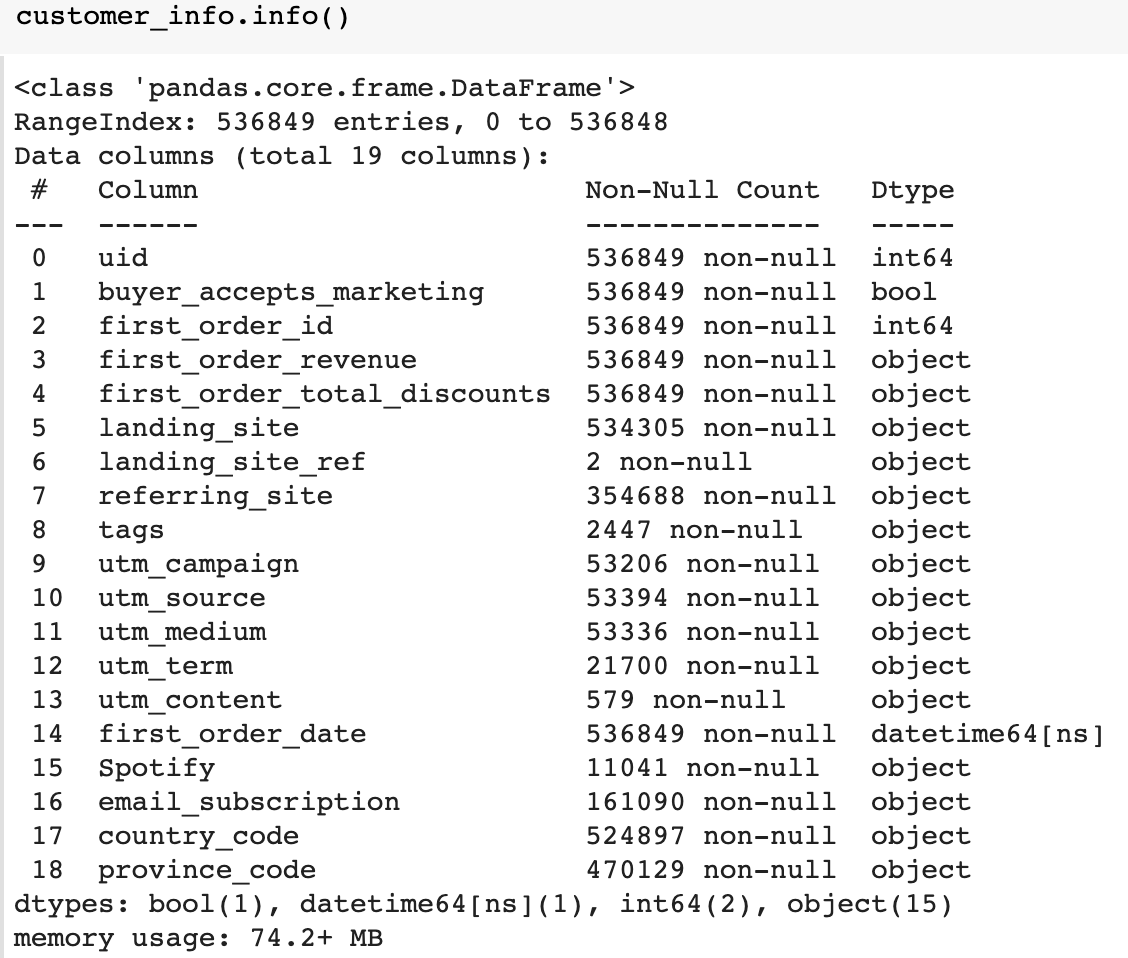
There are three tables:

* **Order information table** contains 840k rows and 15 columns.This table has all customers purchasing order information.
* **Product information table** includes customers’ first purchasing order. This table has 66K rows and 13 columns.
* **Customer information table** has about 0.5MM rows/unique customers and 19 columns. This table also contains customers’ first purchase information.

There is detailed table information.







From the three tables, we have **numerical variables** (such as price and order revenue), **categorical variables** (eg. country\_code), **binary variables** (eg, buyer\_accepts\_marketings), **string variables** (eg product\_title, landing\_site and etc) and **time variable** (eg. date).

Some **special variables** might carry richer information than others.

* Order date: the **holidays or seasonality** can be extracted from. And these information might be useful for modeling
* Country\_code and province\_code are **geo variables**. Some customer location insights can be extracted.
* **NLP variables** such as item\_title, landing\_sites etc. They also contain some product features information and can be extracted via NLP process.

## **2.2. Prepare Data for Consumption:**

In this section, the data wrangling process has been applied to generate a final table for exploratory data analysis and machine learning.

1. Calculate Target Variable

The Target variable has to be created since the raw data source doesn’t provide. Based on the problem definition, a binary target variable should be generated in order to predict the binary classification problem.

Join keys ( uid and order\_id) are used to calculate customers' total order frequency and total revenue during the 12 months since the first purchase for each customer. The target variable is created as ‘repeat\_purchase’, the binary variable. The repeat\_purcase = 1 when total order frequency>1 otherwise repeat\_purchase =1

1. Filter out unuseful rows and columns

Based on the business definition, canceled or refund orders need to be filtered out.

1. Handling missing data

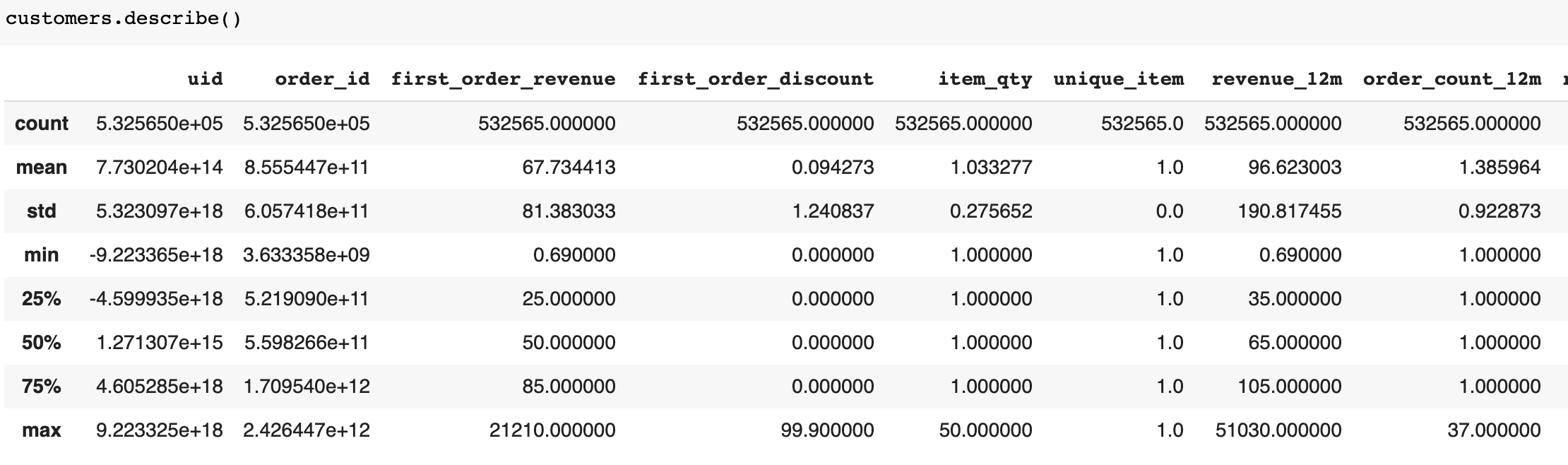
In my data set, only string or categorical variables contain missing data. So for categorical variables, new classes are created to reflect ‘null’ or ‘na’ in these variables. For string variables, they are turned into binary variables based on where the value is missing or not.

Luckily, all missing values are able to be filed based on business definition.

After this process, the final table has been created with 532,565 rows and 21 columns without any missing values in the table. And each row is a unique customer. The target variable is ‘repeat\_purchase’, a binary variable.

## **3. Perform Exploratory Analysis:**

(1) Basic descriptive statistics: min, mean, median, quantiles, max

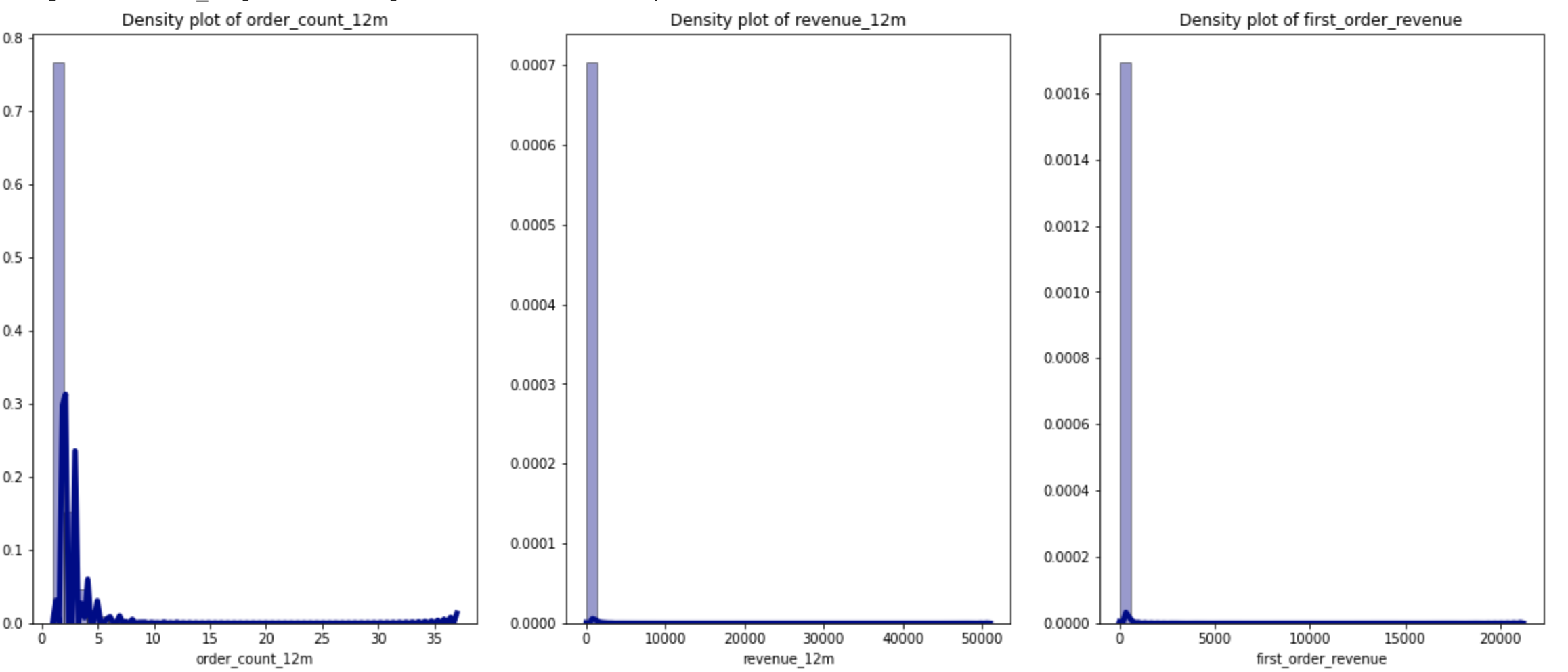


(2) Check distributions

* Bar chart of target variables. Majority of the customers are one-time purchasers.

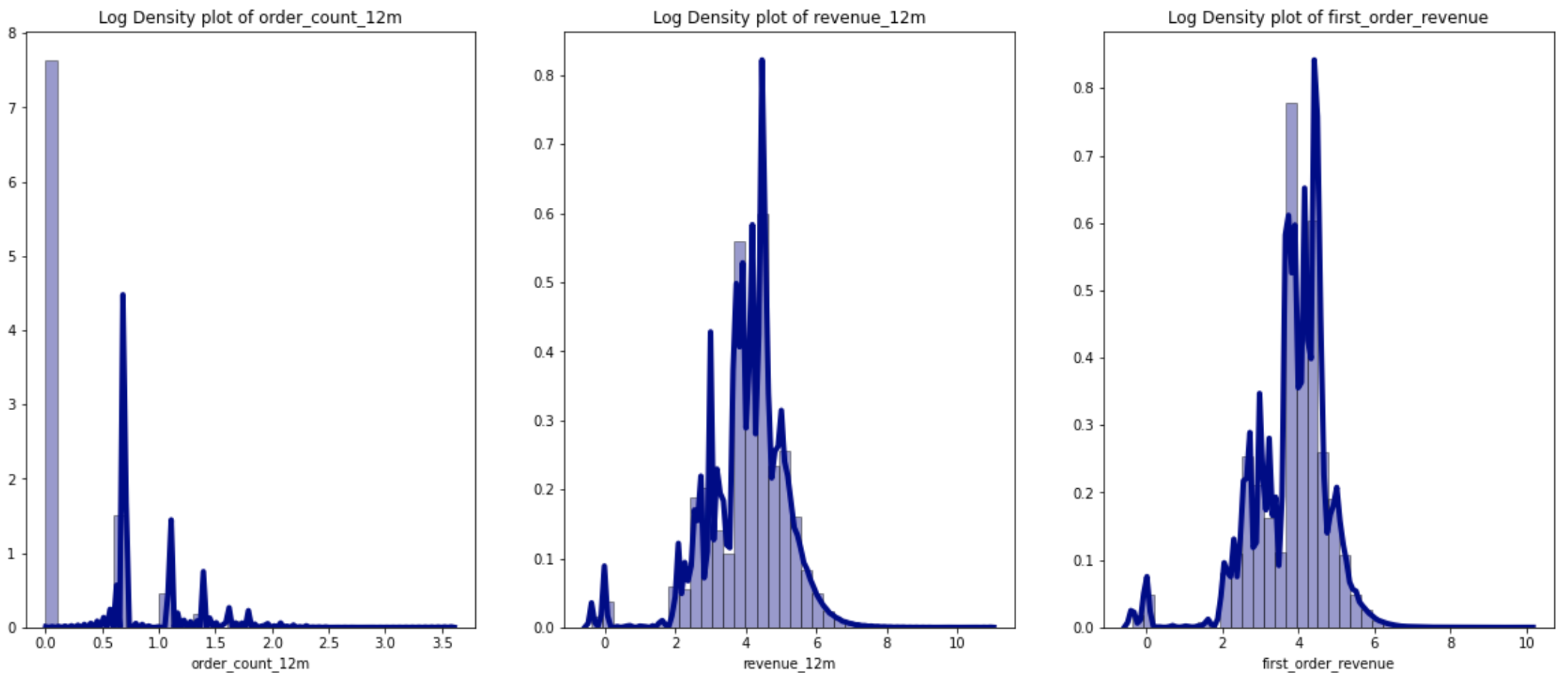


* Generate density plot of orde\_count\_12 (total order frequency per customer), revenue\_12m (total revenue per customer) and first\_order\_revenue.



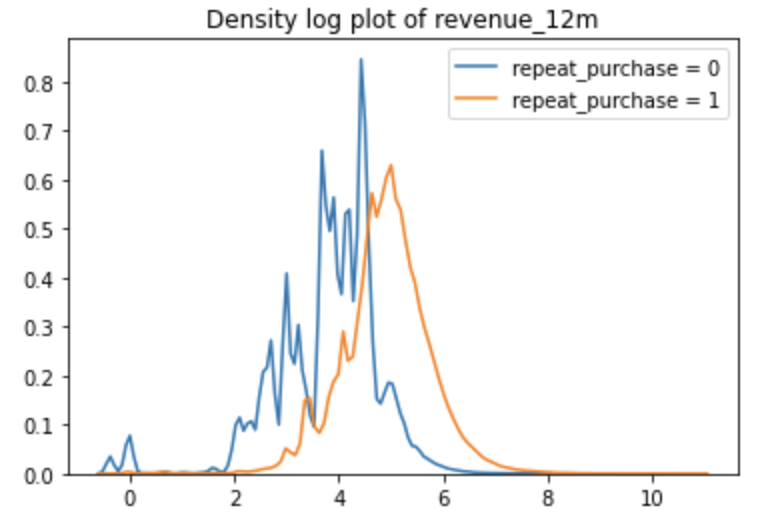
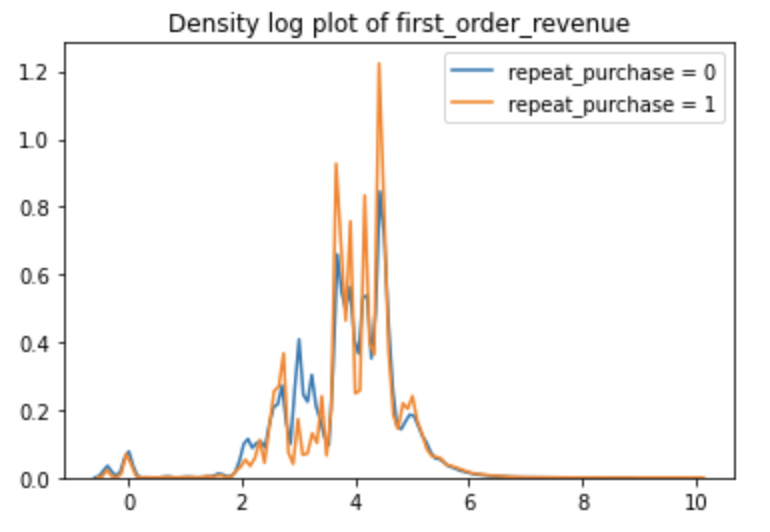
Not surprising, variables are highly skewed especially for monetary variables.

* So let’s look at the features again after log transformation.



First\_order\_revenue and revenue\_12m get closer to gaussian distribution except the little peak on the left by looking at their log density plots above. This tells us that the log transformation helps deal with the skewness for the continuous variables. For some model with gaussian assumption, the log transformation needs to be applied before feeding the data into the model.

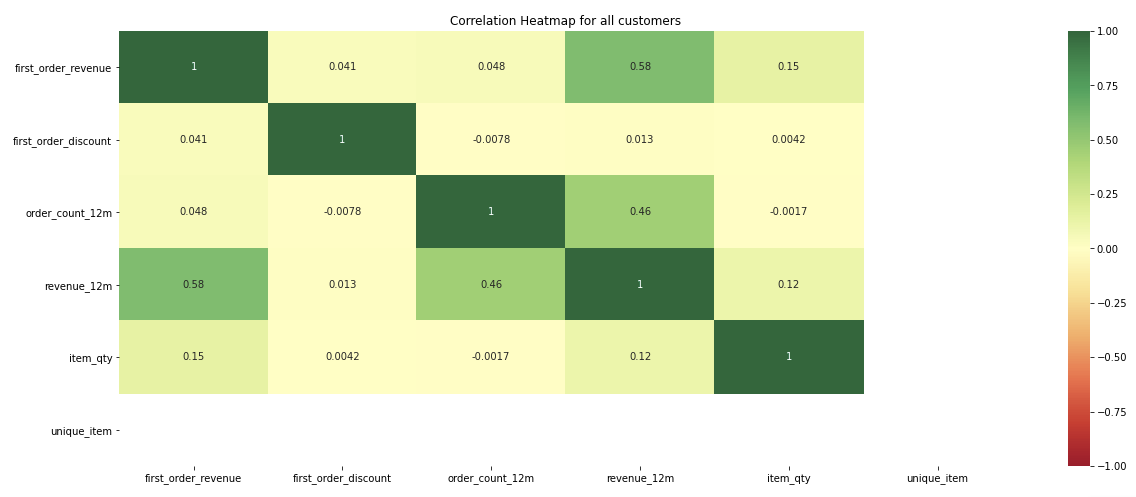
* Let’s check the log distribution between single purchasers and repeating purchasers.



Two types of customers share a very similar distribution shape of first\_order\_reveue. And it seems that the repeat purchasers in general spend more on their purchasers. However, the distribution of revenue\_12 is very different. It turns out the repeat purchasers’ revenue\_12 is almost a gaussian distribution.

From the log plot of first\_order\_revenue, there is a noticeable difference between single purchaser and repeat purchaser where axis value roughly between 2.9 and 3.4. Features can be created to tell models about this information. And in my case, it did improve my models performance.

(3) Correlations



There is a strong correlation between first\_order\_reveue and revenue\_12m and first\_order\_revenue and order\_count\_12m. Since the revenue\_12m and order\_count\_12m are the proxy of the target variable. They will be dropped before modeling in order to prevent leakage. On the other hand, it tells us multicollinearity won’t be an issue when applying a linear model since there is no strong correlation among other variables.

## **5. Feature Engineering**

(1) Create a feature to represent whether the log of first\_revenue\_value is between 2.9 and 3.4 or not.

(2) Create dummy variables for: low cardinal categorical features

(3) Create binary variables for: features with only two classes

(4) Target encoding for high cardinal categorical features via the code below:

**from category\_encoders import TargetEncoder**

**encoder = TargetEncoder()**

**train['country\_code\_encoded'] = encoder.fit\_transform(train['country\_code'], train['repeat\_purchase'])**

**test['country\_code\_encoded'] = encoder.fit\_transform(test['country\_code'], test['repeat\_purchase'])**

**train.drop(columns=['country\_code','province\_code'], inplace= True)**

**test.drop(columns=['country\_code','province\_code'], inplace= True)**

The pro of this method is that it turns high cardinal categorical variables into numerical variables which can be fed into ML models directly. The con is that it represents the training data set information which is very likely to cause overfitting.

What’s the model performance with/o target encoding

## **6. Modeling and hyperparameter tuning**

Three classification models are trained. And Grid Search are applied for hyperparameter tuning.

(1) Logistic Regression

model = LogisticRegression(random\_state=42,class\_weight='balanced')

penalty = ['l2','l1']

solver = ['liblinear','lbfgs']

C = [0.0002, 0.0005,0.0008, 0.000010]

(2) Decision Tree

model = DecisionTreeClassifier(random\_state=42,class\_weight="balanced")

param\_grid = {'criterion': ['gini','entropy'],

'splitter': ['best', 'random'],

'max\_depth': [2,3,4],

'min\_samples\_split': [5, 50],

'min\_samples\_leaf': [2,10,15]

}

(3) RandomForest

model = RandomForestClassifier(random\_state=42,class\_weight="balanced")

param\_grid = {'n\_estimators': [200,300,500,800],

'max\_depth': [3,5],

'min\_samples\_leaf': [10,30],

'min\_samples\_split': [25,50,80]

}

## **7. Evaluation and model selection**

Classification metrics - recall, precision, accuracy, f1 are considered to evaluate model performance and select the best model.

Precision = TP/(TP +FP)

It is implied as the measure of the correctly identified positive cases from all the predicted positive cases. It’s useful when minimizing FP. In other words, FP case costs a lot to the business. In this case, the marketing teams will spend effort and waste marketing budget to the customers who will come back again on their own.

Recall = TP/(TP + FN)

It is the measure of the correctly identified positive cases from all the actual positive cases. It is important when the cost of FN is high. In this case, the marketing teams will miss the chance to market to the people they won’t come and buy again by themselves

Accuracy = (TP + TN)/(TP + TN + FP + FN)

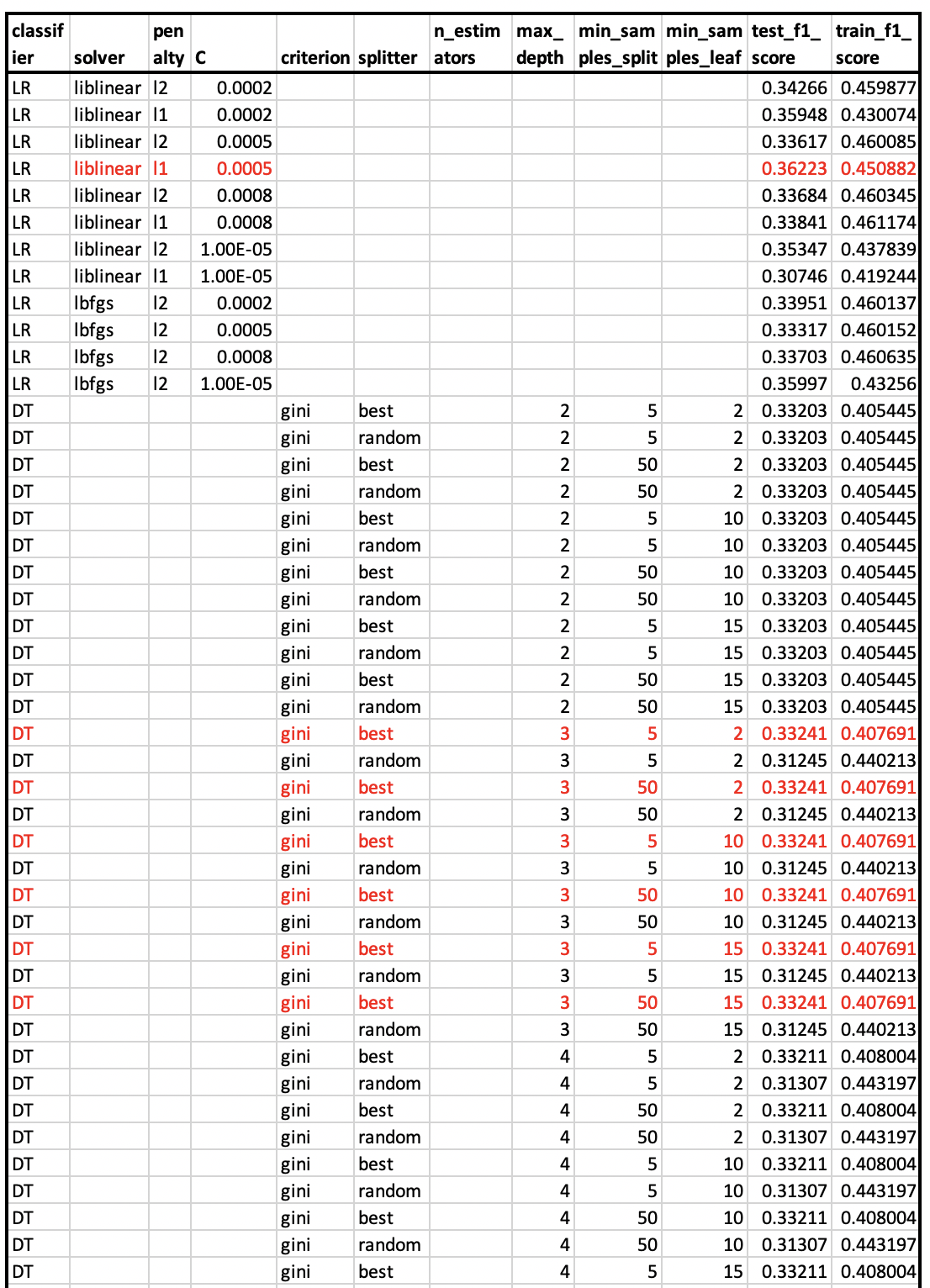
It is the measure of all the correctly identified cases. It is most used when all the classes are equally balanced. However, in this case classes is imbalance.

F1-score = 2 \*(precision \* recall / (precision + recall))

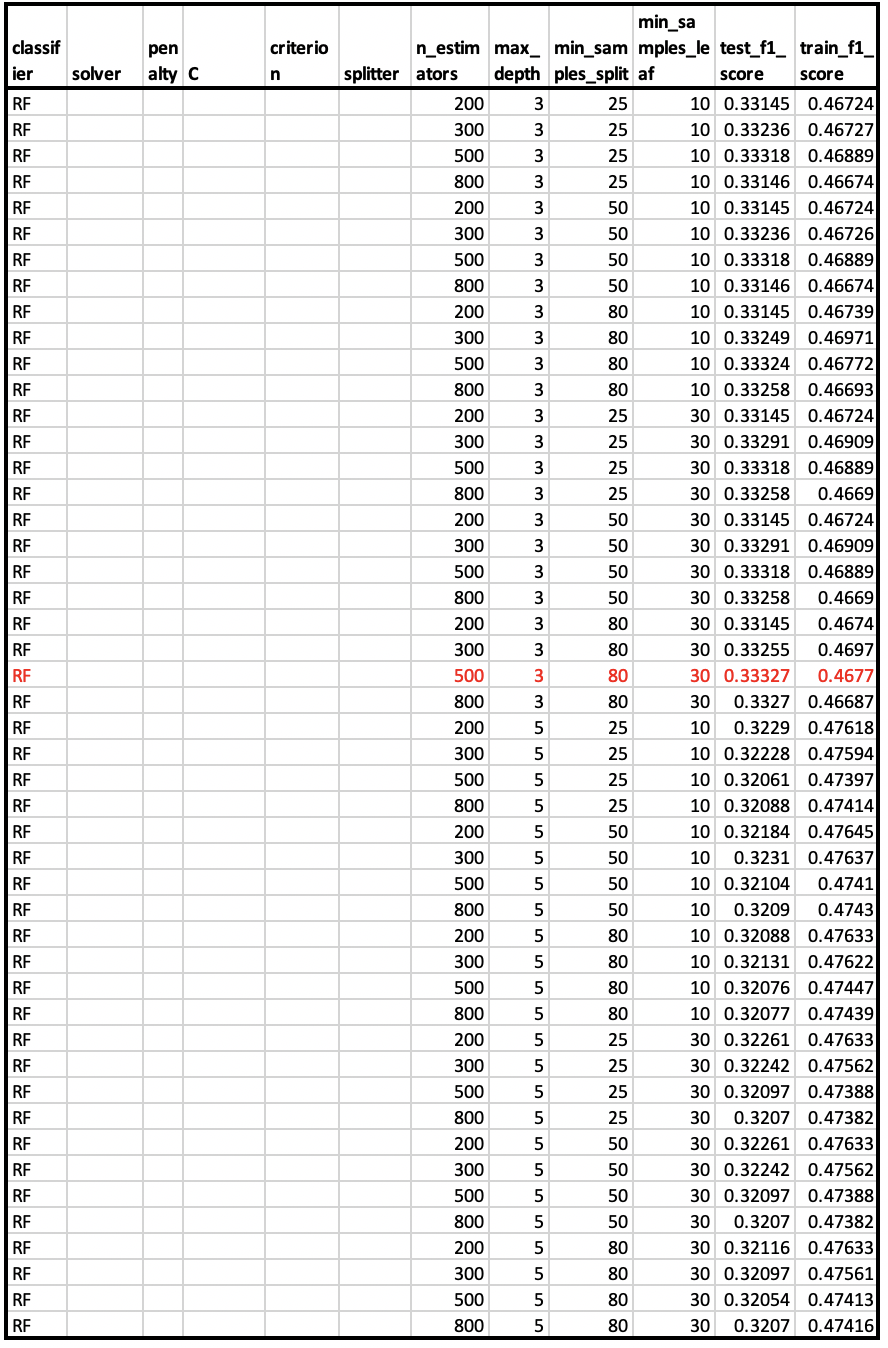
This is the harmonic mean of Precision and Recall and gives a better measure of the incorrectly classified cases than the Accuracy Metric. And It can be used to penalize the extreme values.

We calculate testing F1 score during the hyperparameter tuning. Because for this business case, the marketing team won’t want to waste their marketing budget and at. In other words, there is a trade-off between precision and recall. Therefore F1 score is a good evaluation metric for this case.

Both the testing and train F1 scores are shown in the table below for all combinations of hyperparameters for each classifier. They would help us understand whether the model is overfitting or not.







From the table above, in general the logistic regression classifier is outperformed due to overall higher f1 scores.

Red color highlights the winners within each classifier. And they are:

1. LogisticalRegression

|  |  |
| --- | --- |
| test\_f1\_score | 0.362228044 |
| train\_f1\_score | 0.450882117 |
| Solver | liblinear |
| C | 0.00001 |
| penality | l1 |

1. DecisionTreeClassifier

|  |  |
| --- | --- |
| test\_f1\_score | 0.332406009 |
| train\_f1\_score | 0.407691378 |
| criterion | gini |
| max\_depth | 3 |
| min\_samples\_leaf | 5 |
| min\_samples\_split | 2 |
| splitter | best |

1. RandomForestClassifier

|  |  |
| --- | --- |
| test\_f1\_score | 0.333273089 |
| train\_f1\_score | 0.467701588 |
| max\_depth | 3 |
| min\_samples\_leaf | 30 |
| min\_samples\_split | 80 |
| n\_estimator | 300 |

Comparing testing results from three models :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **classifiers** | **Precision** | **Recall** | **Accuracy** | **AUC** | **test\_f1\_score** | **train\_f1\_score** |
| LogisticRegression | 0.2484 | 0.6684 | 0.5723 | 0.6100 | 0.3622 | 0.4509 |
| DecisionTreeClassifier | 0.2673 | 0.4396 | 0.6793 | 0.5860 | 0.3324 | 0.4076 |
| RandomForestClassifier | 0.2922 | 0.3876 | 0.7182 | 0.5896 | 0.3333 | 0.4677 |

All models are overfit. However, the difference between test score and train score is the opportunity to increase the model performance. Some techniques such as dimension reduction, NLP and etc can be used to increase test score and get the test score closer to the train score.

In this case, the LogisticRegression model has the highest f1 score which could be considered as the best model here.

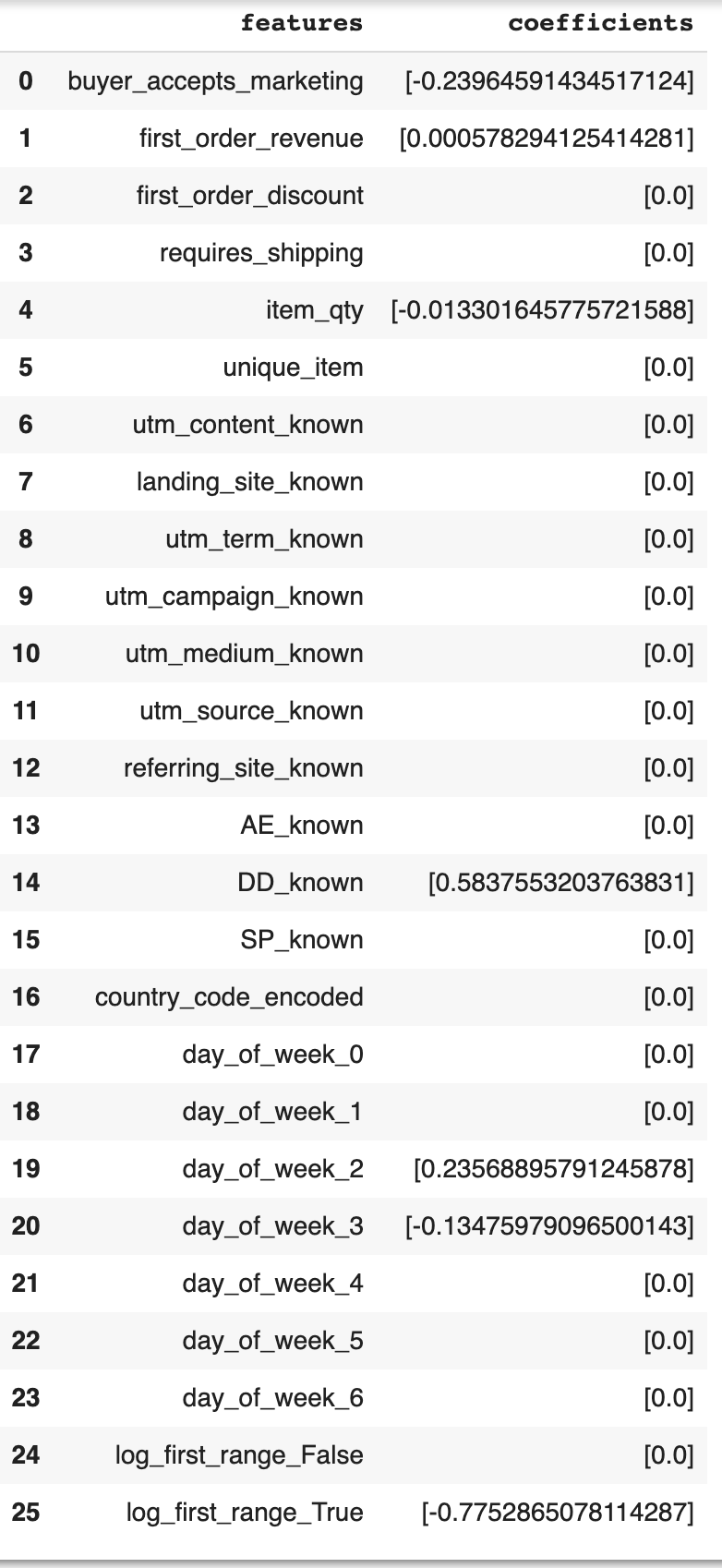
Even though the accuracy looks better than other metrics but in this case looking at accuracy is misleading due to the imbalanced classes. However, all models have low precision, recall, auc and f1 score. This means the models are getting trouble making good predictions. So it’s worth going back to the beginning and considering what other information/data would be added into the model.

## **8. Explanation**

1. Logistic Regression

LogisticRegression(random\_state=42,class\_weight='balanced',penalty = 'l1',C= 0.0005, solver='liblinear')

1. Check the coefficients for each features:



1. The Intercept:



1. Insights:

Overall, all the coefficients are small.

* The coefficient of ‘Dotdigial\_known’ seems significant by comparison to other features. This is not surprising because the active or super fans of the artists will subscribe to the store and get updates from the store when new items are released.
* Looking at the day of week, the coefficient of Wednesday (‘day\_of\_week\_2’) is positive and larger than others. This is an interesting insight which tells us that fans are more likely to purchase on Wednesday.
* Some coefficients are zero because of the L1 penalty. And it also helps for feature selection and we know features with zero coefficients are not important.

(2) Decision tree

1. Plot decision tree



1. General insights:

Our starting point is the first node to the left. Here we have 100% of samples (obviously, we haven’t even started splitting), the proportion between classes is a perfect 50/50 (we balanced the data before building the ‘tree’), and the gini is 0.5, as bad as it can possibly be as a random guess.

The third leaf with the lowest gini index (0.261) among other leaves shows a good segment. But the samples =6%. So this tree model is doing a good job on this small portion of the customers. But it fails on classification on the majority of the customers since the gini index of other leaves is close to 0.5.

The second split feature is log\_first\_range which was discovered during EDA and created during the feature engineering process. Though the gini index is still close to 0.5, log\_first\_range helps build the tree to the third leaf which is a good classification segment. I also tried the decision tree model without this feature and the model was only able to create a good classification of about 2-3% of the sample. Creating useful features is so important, especially if the source dataset is not rich enough.

## **9. Optimize and further work**

Overall, all three models are not able to do a good prediction job due to lack of useful features. In this case, we try to predict customers behavior - whether they will buy again or not. However, we suffer the lack of behavioral data ( eg. touchpoints from any digital service), personality related data ( eg, personal preference) and etc.

Therefore there are some further thoughts:

1. Incorporate digital touchpoints. For example, Google analytics dataset could be considered a good start since it gives out customer’s browsing behaviors.
2. Content based event information. Because the ecommerce stores represent artists' shops. The products are associated with artists songs, concerts or other music related activities. The streaming behavior could be considered a good start.
3. Personality characters information. For example, personal preference regarding product type or music genre and etc.
4. Seasonality: holidays like X’mas, Black Friday, New year and etc. This information can be extracted from the order date variable. They might provide useful information to increase model prediction capability.
5. Dimension reduction techniques such as PCA, LDA can be applied. For example, some specific information such as extracting from products or landing pages etc.
6. Creating new features. Just like the new feature discovered via the EDA process in this case. This process can be replicated when exploring new data sets.