Case study

E-commerce customers satisfaction prediction

Index:

Introduction:

E-commerce is the daily essential of our life now-a-days. E-commerce is the platform where a seller can sell products and customer can buy the products. That means e-commerce platform connects large number of sellers to customers through online. Providing better services to customers is one of the main keys to be successful as an e-commerce seller. How to measure the goodness of the service? The answer is goodness of a service is based on customer satisfaction. If the customers are not satisfied with the services given by the seller, then seller has to focus on service quality to attract the customers, thereby improve the business. So, basically customer feedback plays crucial role in e-commerce business.

Business problem:

In the present system, the e-commerce platform will send a feedback mail to customers after the product is delivered. The customers can give ratings out of 5, also can write down some comments/reviews about the product that he/she has purchased. Using these reviews and ratings, e-commerce platform will rate the products, which helps other people to get the insights about the quality of the product. But according to seller perspective, these reviews will play crucial role to improve the business. But many times, customers would not give any ratings or reviews. How to predict the review score that a customer could give? This is the problem in e-commerce business. Also, the problem can be extended as “Is it possible to predict the review rating that a customer could give before actually he gives the rating?”. If this problem is solved, then it is also possible to predict the rating for which customer had not given any rating. In this case study, my objective is to try to solve this problem, that is Predicting the e-commerce customer satisfaction.

For this case study, I have taken the dataset given by Olist, which is an e-commerce platform in Brazil. Olist connects small businesses all over Brazil to customers with a single contract. Olist has provided over 100k order information that were placed between 2016 to 2018. Similar to all other e-commerce platforms, Olist also send feedback form to customers after the estimated delivery date to get the reviews and ratings. Now, Olist wants to improve the business as well as provide the better service to customers by using the customer satisfaction information. For that it needs to predict the review ratings before the user will give actual ratings. So, my approach is to address this business problem using data science, which is a scientific way to solve this business problem.

ML formulation of business problem:

To solve the business problem using data science, it is needed to pose that problem as classical machine learning problem. First of all, since the data has target variable, it is supervised ML problem. Further we need to predict the satisfaction of customers, that is predicting the ratings. Ratings are discrete ranging from 1 to 5. Hence it is a multi-class classification problem. We have 5 class labels, So, we can treat the problem as 5-class classification ML problem.

Our goal is to predict the rating before the user give the rating/review. Hence, we should not consider data regarding review message, comments, etc as features.

Business constraints:

- There is no strict low latency requirement. But model should not take too much time for predicting, Since we should get the prediction before the user give.

- Low ratings like 1,2,3 are very important with respect to business improvement. So, misclassification of low ratings would cost loss of customers. Hence misclassifications are crucial.
- Since low ratings are crucial, if we get the interpretations of the output, that will be better.

Data overview:

Data link: https://www.kaggle.com/olistbr/brazilian-ecommerce

Data has 9 csv files, namely,

olist_customers_dataset, olist_geolocation_dataset, olist_order_items_dataset, olist_order_payments_dataset, olist_order_reviews_dataset, olist_orders_dataset, olist_products_dataset, olist_sellers_dataset, product_category_name_translation.

Column Name	**Data type**	**Description**
customer_id	object	This is the id of a customer
customer_unique_id	object	This is the unique id for each customer
customer_zip_code_prefix	int64	The is the zip code prefix of customer's location

customer_city	object	This is the name of the customer's city
customer_state	object	This is the name of the customer's state. State name mentioned in short form.

olist_customers_dataset:

olist_geolocation_dataset:

Column Name	**Data type**	**Description**
geo_location_zip_code_prefix	int64	This is the zip code prefix column
geo_location_lat	float64	Latitude of the location
geo_location_lng	float64	Longitude of the location

geo_location_city	object	Name of the city
geo_location_state	object	Name of the state

olist_order_items_dataset:

Column name	**Data type**	**Description**
order_id	object	This is the id for each order
order_item_id	int64	This is the id for ordered item, ranging from 1 to 21
product_id	object	This is the id for each product
seller_id	object	This is the id for each seller
shipping_limit_date	object	Limit date for shipping
price	float64	Price of the product

freight_value	float64	This is the transport charges for the order

olist_order_payments_dataset:

Column Name	**Data type**	**Description**
order_id	object	Id of the order, all the below details are corresponding to this order_id
payment_sequential	int64	Payment sequential details
payment_type	object	Type of the payment (credit card, ballot)
payment_installation	int64	Number of installations
payment_value	float64	Value of the payment

olist_order_reviews_dataset:

Column name	Data type	Description
review_id	object	Id for the review
order_id	object	Order id for which the reviews are made
review_score	int64	This is the rating score, ranging from 1 to 5
review_comment_title	object	Title of the review comment
review_comment_message	object	This is the detailed review about the order
review_creation_date	object	Date time on which the review mail sent to user
review_answer_timestamp	object	Timestamp when the user give the review

olist_orders_dataset:

order_id	object	Id of the order

customer_id	object	Id for the customer
order_status	object	Delivery status of the order
order_purchase_timestamp	object	Time stamp of the purchase
order_approved_at	object	Date time when the order is approved
order_delivered_carrier_date	object	Date time when the order is delivered to logistic partners from the seller
order_delivered_customer_date	object	Date time when the delivery happened
order_estimated_delivery_date	object	This is the estimated delivery date

olist_products_dataset:

product_id	object	Id of the product
product_category_name	object	Name of the category to which the product belongs to
product_name_lenght	float64	This is the length of the name of the product
product_description_lenght	float64	Length of the description of the product
product_photos_qty	float64	Number of photos that the product has
product_weight_g	float64	Weight of the product in grams
product_length_cm	float64	Length of the product in centimetres
product_height_cm	float64	Height of the product in centimetres
product_width_cm	float64	Width of the product in centimetres

olist_sellers_dataset:

Column name	Data type	Description
seller_id	object	Id of the seller
seller_zip_code_prefix	int64	Prefix of the zip code of the seller's location
seller_city	object	City name of the seller
seller_state	object	State code of the seller

product_category_name_translation:

Column name	Data type	Description
product_category_name	object	Name of the product category in Portuguese language
product_category_name_english	object	Name of the product category in english

Performance Metrics: (possible)

- Multi-class confusion matrix
- Macro F1 score
- Precision Recall curve for each class
- Multi-class log loss
- Balanced accuracy score

Research-Papers/Solutions/Architectures/Kernels:

- **Existing solution**: https://www.kaggle.com/andresionek/predicting-customer-satisfaction

This is the kernel on Kaggle given by the contributor of the dataset. This is the existing solution to our problem. In this kernel, he considers only the data till the product delivered to the customers as features. The problem is framed as a regression problem, using MSE as metric. More importantly, it consists nice feature engineering part. Features like,

"Working days estimated delivery",
"Working days actual delivery time",
"Difference between actual and estimated delivery time",
"Average product value", "Order freight ratio", "Total order value", "Is late", etc.

He used linear regression, Random Forest Regressor models and got 0.53 as minimum MSE. He kept further process for experimentation.

The main use of this kernel for the present case study is feature engineering. We can experiment with use these interesting features.

- **Repeat buyer prediction for e-commerce**
  https://www.kdd.org/kdd2016/papers/files/adf0160-liuA.pdf

This is a paper is a winning solution on prediction of repeat buyers, which was a competition problem at IJCAI in 2015, hosted by Alibaba. The author discussed about the key factors which affect the customer for repeated buying. The data contains user information, merchant information. Using these data, they engineered very complex features like merchant market share on brand, merchant-user share on brand, that is market share features, Also, similarity features between user and merchant, age/gender related features, etc. Also, they presented the feature rankings using XG Boost model.

From this we can see that, user-seller similarity features, user-brand aggregation features, seller-brand aggregation features those are important features. In our study, we are predicting the satisfaction of users. If a user is satisfied then there are high chances of repetition of buying. Hence, we can use some of the features that are discussed in this paper for experimentation.

Some interesting features that could be experimented in our case study are:

Market share features: * merchant's market share on a brand, merchant's user share on a brand, brand's market share within merchant, brand's user share within merchant.

User merchant similarity: It can be calculated using market share features.

So, this paper would help us by giving some interesting feature ideas to solve our problem.

- **LightGBM Classifier**: https://papers.nips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

This is the paper of LightGBM which is the modification of GBDT and a light version. GBDT is computationally very expensive when we have large data and large number of features. GBDT uses all the features for calculating the information gain of all possible split point, by this its computational complexity is proportional to number of features and number of data points. Whereas LGBM is a modification to GBDT by considering Gradient Based One Side Sampling (GOSS) and Exclusive Feature Bundling.

GOSS is a technique where the model keeps all data points with large gradients and randomly sample data points with small gradients. Data points with small gradients are contributing less to the loss that means they are well trained points. Totally dropping all the points causes change in distribution of data. So, using GOSS, model will keep all the points with high gradients and randomly sample data points with small gradients. For this model will have some threshold to select the gradients. So, by this model will give weight to data points like in case of Adaboost which is not available in GBDT.

LGBM grows trees leaf-wise. Main improvement in LGBM is histogram-based algorithm in case of continuous feature. Each continuous feature is bucketed into discrete bins. Now in order to compute the best split, model will only iterate through each bins instead of each point. For categorical features with one hot encoding will create sparse feature space. And the exclusive feature bundling will take care of this in LGBM.

In our case study we have large number of data points as well as we have more categorical features, which after one-hot encoding creates large number of sparse features. So, using LGBM we can handle this issue as discussed in the paper. LGBM implementation is available in https://lightgbm.readthedocs.io/en/latest/index.html . Also here they mentioned that multi class log loss can be used with LGBM. So, we can experiment with LGBM in our case study.

- **SMOTE technique**: https://arxiv.org/pdf/1106.1813.pdf
  In this paper SMOTE technique is discussed. When the data is imbalanced there are two types of data level modifications, one is undersampling another on is oversampling. Undersampling means sample the majority class points to make the balance between minority and majority class labels. Oversampling means randomly sample the minority class points that means randomly repeat the minority class points to balance. But these two approaches have their own disadvantages, like oversampling cause overfitting and in undersampling we are throwing out the data points which is not a good idea.
  SMOTE is a technique of creating synthetic points in the region of minority class points. This is also oversampling but not randomly repeating the points. For this, SMOTE uses KNN algorithm, and it depend on percentage of oversampling required. If we need 300% oversampling, then we take 3-nearest neighbours to each minority points in the minority space, and calculate the length of line joining the point to 3- nearest neighbours and randomly multiply that distance with a number between 0 and 1. The new point is considered in that result of multiplication.
  In our case study, data is imbalanced. To balance the class in the data level modification we can use SMOTE.
  SMOTE is implemented in python imbalanced-learn library.
  https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

- **Blog on undersampling, oversampling, SMOTE, ensemble models**: https://xang1234.github.io/louvain/

This blog contains techniques to deal with imbalanced data. Here, the author mentioned about data level handling like undersampling, oversampling. Also, synthetic data creation method SMOTE. He suggested to try with different algorithms which are actually designed for handling class imbalance, like, balanced bagging classifiers.

We can consider this blog, because we have imbalanced data. And we can experiment with the techniques like balanced bagging classifiers, which are algorithmic level handling of imbalanced data. For this we can use *imbalanced-learn* library, which is mentioned in this blog.

- **RUSBoost algorithm**: https://www.researchgate.net/publication/224608502_RUSBoost_A_Hybrid_Approach_to_Alleviating_Class_Imbalance

This is a research paper on RUSBoost algorithm. The author explains RUSBoost with its algorithm. RUSBoost is a robust classifier for skewness of data, which is a combination of sampling and boosting. This is mainly useful for imbalanced data. Also the author gave experimented results on comparing SMOTEBoost and RUSBoost. According to the author RUSBoost performs better compared to SMOTEBoost and speed and simple. SMOTEBoost uses smart oversampling technique, RUSBoost will achieve the same goal by using random undersampling.

RUSBoost will perform random undersampling at each step of iteration in boosting algorithm.

Since we have skewed data, we can experiment with this model, imbalanced-learn library will provide implementation of RUSBoost model.

- https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/

This blog explains about undersampling and oversampling and combined approach. This is very needed in our case study since we have imbalanced data.

Modest oversampling to minor class and modest undersampling for majority class is discussed in this blog. We can try this approach, since only undersampling and oversampling are not very good approachs.

- https://www.researchgate.net/publication/323111412_The_effects_of_customer_satisfaction_with_e-commerce_system

This is a paper on effect of e-commerce customer satisfaction on e-commerce. In this paper the author discussed about the key factors which has effect on customer satisfaction. This paper helped me for understanding the customer and e-commerce behaviours. Customer satisfaction is mainly affected by:

Service delivery, that is service should be delivered in time to get the best reviews.

Demographic information like age, gender, employment status, those affect the ratings. But we don't have that information about the users in our collected data.

When user uses credit card, the amount of fees and other charges also has effect on customer satisfaction. If this charge is high then customer could be unsatisfied.

These are the information which help us while doing EDA.

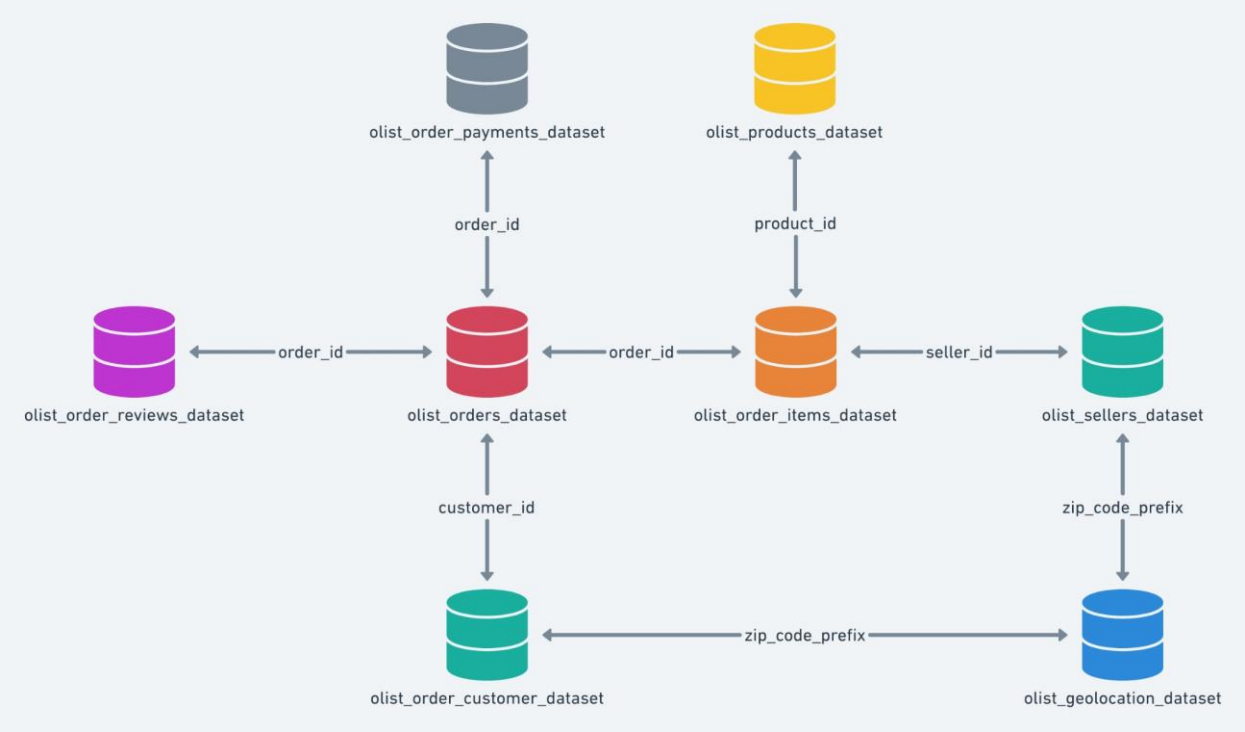
- https://towardsdatascience.com/using-data-science-to-predict-negative-customer-reviews-2abbdfbf3d82
  This is a blog, where the author considers the problem as binary classification problem, and his main intention is to find negative reviews. Because negative reviews cost more for the business. The approaches in this blog are helpful for analysing low ratings. Also, the author gave the result that bad reviews are sensitive to late deliveries. Though this is common sensical, we shall check this in our case study.

- https://www.kaggle.com/goldendime/data-cleaning-viz-and-stat-analysis-on-e-com
  https://www.kaggle.com/jsaguiar/e-commerce-exploratory-analysis

These two kernels are very helpful for EDA. These kernels contain detailed EDA and visualizations of features. Important takeaway from these kernels is time series analysis of sales and product values over the time. Also the geolocation analysis of sellers and customers would help us while doing EDA.

First Cut Approach:

1. Download the data from kaggle, which has 9 csv files.
2. There is one file translation.csv, this file is not needed for the analysis part. Hence except this file, we have to merge all the files
   as described in the data schema. Let us consider this as final data.



3. The translation.csv has English version of category names. So we can translate the category name in the final data using translation.csv file.

4. Target variable is review_rating which has the discrete values between 1 to 5. If we consider multiclass classification, it becomes a 5-class classification problem. [We can also convert it to binary by considering rating less than or equal to 3 as negative (not satisfied) and greater than 3 as positive(satisfied).] Let us consider this as a 5-class classification problem.
5. **Choosing metric:** I have chosen some metrics which could be better to use for this problem. They are,
    - Multi-class Confusion matrix: We can easily get the number of misclassifications. Also, easy to understand in 5-class classification.
    - Micro F1 score: It is better to use micro F1 score, since we have class imbalance. Micro F1 score will give weighted score.
    - Balanced accuracy: This is nothing but taking average of recall of each class.
- Precision Recall curve for each class: This could be useful to check precision and recall for each class. It will be cumbersome to handle multiple curves, but let us consider this now. If not needed then we can drop this.
- Multi-class log loss: If we get the probabilistic output, it is easier to use log loss. For multi-class setting we can use multi-class log loss.

    These are the possible metrics.

6. **Data overview:** This step includes checking the shape of the dataframe, and we have to analyse each column of the dataset, and get the high-level info about each feature, like data type.

7. **Data pre-processing**:

- **Handling null values**: In this step, we have to check for null values. There are some possibilities, **very less number of null values**: If the percentage of null values are very small, then we can drop those data points. **Null values which cannot be imputed:** Some null values like datetime

features cannot be imputed, So, if we have null values in datetime features we have to drop those datapoints. **Null values in numerical features**: If we have null values in numerical features, we can try mean, median imputation. Or we can try to model linear regression with the most correlated feature taking missing value feature as target variable to impute. **Categorical features:** In case of categorical data, we can treat null values as on of the categories.

- **Removing duplicates:** There are possibilities of duplicate rows in the data. We should check that. If there exist any duplicate rows in the data, we should remove those duplicate filters.
- **Convert data type if necessary:** We have date time features in the data, they are in the object type. We have to convert that to datetime format to handle datetime easily.
- **Removing the features:** In this step we are going to drop the features which are related to review data. That is review comment, review message, review creation datetime. Because we are going to predict the rating without these information (Before the user giving these information)

8. Now check for the class label distribution. According to the study, e-commerce customer review rating mostly distributed in J type shape. That is 5-star is very high followed by 4-star and 1-star. 2-star and 3-star ratings will be very less. We have to check the class distribution, by this we get the idea about class imbalance. I observed that 5-star > 4-star > 1-star > 3-star > 2-star. 2-star data points are very less. So, we have imbalanced/highly skewed data. This is one problem we have to address in the further steps.

9. **Basic EDA:** This step includes analysis of each existing features.

- Simple statistics about the features: We can get about the mean/median, standard deviation, 25^{th} and 75^{th} percentiles of numerical features. Similarly mode, less frequent categories these types of statistics for categorical features.
- Also we can answer some questions like, What are the top categories, In which state there are high sales, Which payment type is more frequent, What number

of payment installation is common, These type of questions are important to get the clear idea about the data.

- We can plot box plots for getting the idea of outliers. If there are we can remove those outliers by filtering those values. This is based on the insight that we get from the boxplot analysis.
- Correlation of features with the target variable: First we shall check the correlation of features with the target variable to get the idea about which feature has more driving nature to predict ratings.
- Univariate analysis: We shall carryout univariate analysis of each feature corresponding to each class. We have to check whether any feature alone able to classify class labels or not. We can do this by simply plotting pdf of feature for each class. If there is clear separation between class labels that is good. Also we can plot cdf to check how classes are separated below some percentile.
- Bivariate analysis: In this step we have to analyse by taking 2 features at a time. How pair of features could help us to separate class labels is the main objective of this analysis.

10. **Feature engineering:** This is the very important step, because by creating good features we can get the best result from the model. These are some features that I want to experiment:

- Difference between actual delivery time and estimated delivery time
- Late_or_early : This is based on the previous feature. If the difference is negative then that is early, if difference is positive then that is late. We can consider this feature as binary feature. Mostly customers feel bad when the delivery is late.
- Distance between customer's location and seller's location. This can be calculated using latitude and longitude information. If the distance is small, then delivery will be quick consider to farther sellers. This feature we can try.
- Difference between date of purchase and date of estimated delivery. If this difference is large then customers could feel unhappy.
- Difference between data of purchase and date of approve. Large difference cause delay of delivery and it could lead to customer's unsatisfaction.

- We can calculate the average speed of carrier by using (datetime difference between carrier_delivered_data and customer delivered_data) divided by the distance of the seller to customer.
- Along with these features we can create advanced features that are discussed in research paper section.

11**. EDA of new features:** All the features that we have created might not be useful. So, we have to analyse those features based on the class separation. Also, we can check the correlation of the new features with the rating.

12. The new features that we discussed can be created before train test split. But the advanced features that are discussed in research paper section should be calculated after train test split to avoid data leakage.

13. **Train test split**: The total data should be split into train, cross validation and test. We should do stratified splitting. We can do 80:20 split. After splitting we can check the distribution of class labels in each set. Distribution should be same.

14**. Featurization**: In this step we have to normalize/standardize the numeric features. And we can perform one-hot encoding/response encoding on categorical data. (This should be done before handling imbalanced data because techniques like SMOTE requires KNN model to run.)

15. **Handling imbalance in train data**: Since we have imbalanced data, we have to consider some strategy to overcome this. These techniques should only be applied to train set. We can experiment with data level modifications like oversampling or SMOTE. Oversampling could give overfitting problem. We have to experiment with these techniques. Since we are throwing out data points in undersampling, this is not a good idea to choose.

16. After trying with data level modifications we could try algorithm level modifications, that will be discussed in the coming points.

17. **Model building:** Now after step 15, we have balanced train data. Now we have to experiment with different classification models. Since we have large data, the first preference that I have is logistic regression.

18. **Hyper parameter tuning**: We have to do hyperparameter tuning corresponding to each model that we choose using cross validation data. After cross validation we can use best hyper parameters to train and test on the test data to get the actual result.

19. We can try with simple models like Logistic regression, SVM, KNN, etc.

20. in the next step we can use ensemble models like Random Forest, RUSBoost, XGBoost, LightGBM. By experiment with these complex models, we can compare simple models and these complex models.

21. The next experiment is with the algorithmic level modification for imbalanced data. For that I want to experiment with RUSBoost, and some modifications of these Bagging and Boosting algorithms for imbalanced data which are available in imbalanced-learn library.

These are the initial steps that I want to follow. After this, based on the results that I get, I can choose further experiments and further study.