I. Definition

Project Overview

The current project is about how to determine the level of clients' satisfaction and what parameters affects it to a greater extent. The restaurant business is taken as an example.
The data for the project is taken from Kaggle "Restaurant Data with Consumer Ratings" (https://www.kaggle.com/uciml/restaurant-data-with-consumer-ratings).
The input data has details about 138 users which rated 130 restaurants. The details of restaurants are provided as well.
Several models have been built to predict what characteristics of restaurants are dominant in the assessment and in what category - nice, good or satisfied - a restaurant falls into.
It might help a restaurant's owner to correct deficiencies and improve strengths of business.
The restaurant business has been taken as it's always on demand. In addition, this project can be used as basis for the analysis of other services with different variables.

Problem Statement

The purpose of the project is to build the model which will define whether a client likes a restaurant or not. It will be based on the restaurant's characteristics. The first step is to clean and analyze the given data. Some details are redundant and cannot help to build a model. The lack of details for some restaurants will be taken into consideration as well.
During the first step, I explored the data and tried to find some relations between restaurants and users data.
This step helped me to understand how the data should be encoded and what algorithms are to be used to make a prediction when dealing with categorical input and output. I used the mix of different algorithms to get the result. For more accurate predictions, I divided restaurant into few clusters and selected features when running a model. Clustering and features selection were conducted independently.
The algorithms for clustering and features selection:

- K-means
- Backward elimination
- Chi2

The algorithms for a prediction:

- Random Forest Classifier
- Multiclass classifier from Linear Learner from Amazon SageMarker
- Balance Classifier

The data was splitted into the training and test sets to see how good or bad the model is.

Metrics

For measuring model performance, the following metrics will be used.

- Confusion matrix. I think this method is a good approach to evaluate the performance of the classification algorithm. Confusion matrix is the most commonly used metrics for such types of classification algorithms where output can be two or more classes.

True Positive (TP)

		Prediction				
		C_1	C_2	C_3	...	C_n
Actual	C_1	TP_1	FN_{12}	FN_{13}	...	FN_{1n}
	C_2	FN_{21}	TP_{22}	FN_{23}	...	FN_{2n}
	C_3	FN_{31}	FN_{32}	TP_3	...	FN_{3n}
	...	...	...	...	...	...
	C_n	FN_{n1}	FN_{n2}	FN_{n3}	...	TP_{nn}

False Negative (FN)

		Prediction				
		C_1	C_2	C_3	...	C_n
Actual	C_1	TP_1	FN_{12}	FN_{13}	...	FN_{1n}
	C_2	FN_{21}	TP_2	FN_{23}	...	FN_{2n}
	C_3	FN_{31}	FN_{32}	TP_3	...	FN_{3n}
	...	...	...	...	...	...
	C_n	FN_{n1}	FN_{n2}	FN_{n3}	...	TP_{nn}

False Positive (FP)

		Prediction				
		C_1	C_2	C_3	...	C_n
Actual	C_1	TP_1	FN_{12}	FN_{13}	...	FN_{1n}
	C_2	FP_{21}	TP_{22}	FN_{23}	...	FN_{2n}
	C_3	FP_{31}	FN_{32}	TP_3	...	FN_{3n}
	...	...	...	...	...	...
	C_n	FP_{n1}	FN_{n2}	FN_{n3}	...	TP_{nn}

True Negative (TN)

		Prediction				
		C_1	C_2	C_3	...	C_n
Actual	C_1	TP_1	FN_{12}	FN_{13}	...	FN_{1n}
	C_2	FN_{21}	TP_{22}	FN_{23}	...	FN_{2n}
	C_3	FN_{31}	FN_{32}	TP_3	...	FN_{3n}
	...	...	...	...	...	...
	C_n	FN_{n1}	FN_{n2}	FN_{n3}	...	TP_{nn}

- Precision for each class indicating the proportion of predicted outputs referring to class label and the actual number of outputs of this label. In other words, it measures the number of correctly predicted classes. Precision is equal to ratio of correctness in the data points classified as positive.
- Recall for each class indicating the proportion of actual class which was correctly classified. Recall measures ratio of data points that were classified as positive among all positive data points.
- Accuracy is measured as a percentage and indicates how many classes were predicted correctly.

For calculation these metrics following formulas are used:

$$Precision = \frac{1}{n}\sum_{i=1}^{n}\frac{tp_i}{tp_i + fp_i}$$

$$Recall = \frac{1}{n}\sum_{i=1}^{n}\frac{tp_i}{tp_i + fn_i}$$

II. Analysis

Data Exploration and Visualization

Those two segments were combined as closely intertwined with each other. I took few examples of graphs from Jupyter Notebook to display the data.

The data for this project was taken from ‘Restaurant Data with Consumer Ratings’. There are nine .csv files which can be divided into three groups: 1) restaurants related data ; 2) users related data; 3) ratings given by users.

Files related to restaurants:

1. geoplaces.csv stores the general data about restaurants. There are 20 self explanatory columns in the file.
    1.1. placeID;
    1.2. latitude;
    1.3. longitude;
    1.4. the_geom_meter;
    1.5. name;address;
    1.6. city;
    1.7. state;
    1.8. country;
    1.9. fax;
    1.10. zip;
    1.11. alcohol;
    1.12. smoking_area;
    1.13. dress_code;
    1.14. accessibility;
    1.15. price;
    1.16. url;
    1.17. Rambience;
    1.18. franchise;
    1.19. area;
    1.20. other_services
2. chefmozaccepts.csv stores payment options of restaurants.
3. chefmozcuisine.csv stores restaurants’ cuisines.
4. chefmozhours.csv stores restaurants’ working hours.
5. chefmozparking.csv stores parking options of restaurants.

Files related to users:

1. userprofile.csv stores the general data about users. There are 19 self explanatory columns in the file.
    1.1. userID;
    1.2. latitude;
    1.3. longitude;

1.4. smoker;
1.5. drink_level;
1.6. dress_preference;
1.7. ambience;
1.8. transport;
1.9. marital_status;
1.10. hijos (meaning children from Spanish);
1.11. birth_year;
1.12. interest;
1.13. personality;
1.14. religion;
1.15. activity;
1.16. color;
1.17. weight;
1.18. budget;
1.19. height

2. usercuisine.csv stores restaurants' cuisines which users evaluated.
3. userpayment.csv stores payment options of users.

All files related to restaurants and users have the common value in datasets: placeID for restaurants and userID for users. Those values will be used as a key value when merging the data. The last file rating_final.csv stores placeID, userID, and three types of ratings given by users: rating, food_rating, service_rating.
All files are saved in the folder and then will be used when exploring and visualising the data.

The first step is to upload the data into Jupyter Notebook and then save all files in the correct format as the initial files have as mixed types of delimiters.
Before analyzing the data, all values in the datasets were lowercased in case same values are written in different ways, which was the case for restaurants' payment methods. Then, all columns' values were checked and some values were rearranged to decrease the number of options in one column.
Some examples of values' updates.

1) Different ways of writing. Values 'slp', 'san luis potosi', 's.l.p.', 'san luis potos' from the column 'state' all refer to one state 'San Luis Potose'. They were replaced by 'san luis potosi'.
2) Similar meanings. There are 5 different options in smoking_area: 'none', 'not permitted', 'only at bar', 'section', 'permitted'. First two values were combined into 'not permitted'; the remaining three values were combined into 'permitted'.

All datasets were checked and the required changes were done for all values. In case some options were reassigned or merged into one group, it was done basing on a subjective opinion of a developer and her logical intuition.
During the analysis, columns which values are irrelevant, e.g. zip/url which are mostly empty, or difficult to evaluate, e.g. latitude/longitude, were dropped. Moreover, the dataset of restaurants working hours was not considered during the further analysis. There are 273 different shifts over three types of working days. Many working hours are intersected with

each other. As there are many different options of working hours of restaurants, it's been decided to disregard this dataset. If a client evaluated a restaurant, it assumes he attended it and the working shift is not an issue for the client. In addition,the data does not state what hours are more profitable for a restaurant and it's not clear how it can be evaluated.

placeID			
days	mon;tue;wed;thu;fri	sat	sun
hours			
00:00-00:00	24.0	27.0	49.0
00:00-23:30	213.0	225.0	243.0
01:00-01:00	NaN	1.0	NaN
01:00-20:30	1.0	NaN	NaN
01:00-23:30	NaN	1.0	1.0
...	...	...	...
21:30-13:00	2.0	2.0	2.0
21:30-19:00	2.0	NaN	NaN
21:30-21:00	2.0	2.0	2.0
21:30-22:00	2.0	2.0	2.0
21:30-23:00	2.0	2.0	NaN

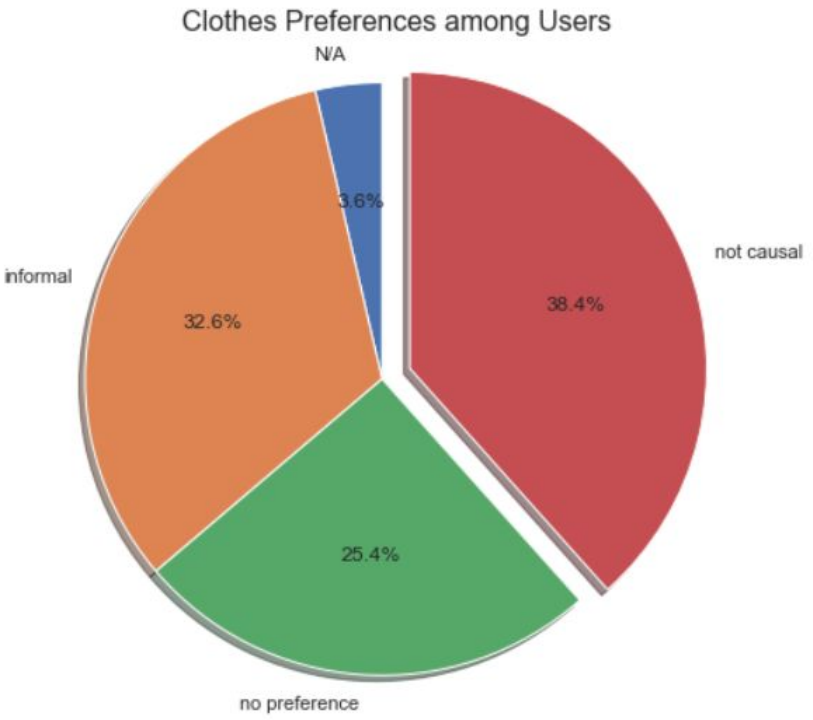
273 rows × 3 columns

Basing on the restaurants and users data, it’s displayed how restaurants are divided into different groups per each category and what group of users is dominant in users’ categories. The restaurants’ graphs display how many restaurants occur in each group. It can be used to check restaurants with the most frequent features and align your own strategy when running the business.

Restaurants graph’s example: Parking Options of 130 restaurants.

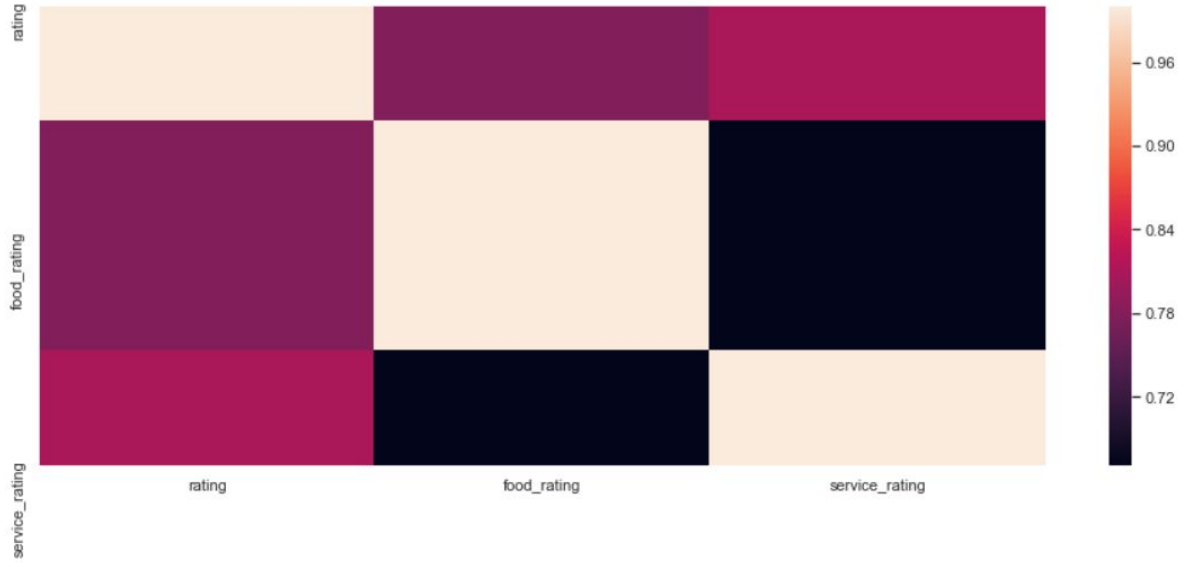
The users' graphs display users preferences and how many users occur in a group. A restaurant's owner might change some restaurant's features if the majority of clients fall into a specific category and the owner wants to target this group of clients.

Users graph's example: Clothes Preferences among 138 clients.



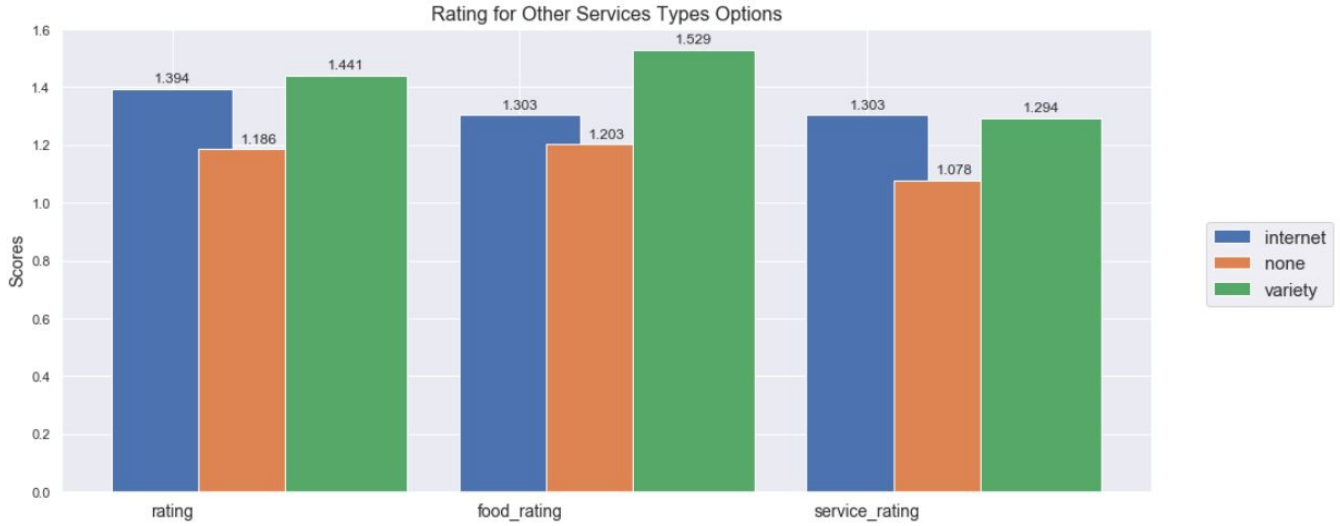
During the data analysis, the correlation between three types of ratings from the rating_final.csv was checked.

```
The rating correlation:
                rating  food_rating  service_rating
rating          1.000000     0.779939        0.809132
food_rating     0.779939     1.000000        0.670809
service_rating  0.809132     0.670809        1.000000
```



Lastly, the average rating per four different categories (franchise, area, rambience, other_services) from r_general as those categories were selected as features which affect most the rating of restaurants. Its selection will be explained in details in the section 'methodology'.

Example: Rating for Other Services Types Options



Algorithms and Techniques

After analyzing the data, I decided to work with the restaurants data and ratings. Basing on the restaurant features, I will predict the rating of restaurants.
As I mention in the ‘Data Exploration’ Section, restaurants have many characteristics which are not relevant and hard to predict. Even after removing some columns, e.g. country, address, zip, etc., there are still many input variables. I decided that some features are more significant from the point of view of customers which affect their ratings. In the beginning, I used a basic linear learner with multiclass classifier to predict ratings but the accuracy was really low, around 52.5%. So I was convinced there is indeed too much noise caused by the number of inputs.
To solve this issue, I have to select top features which affect most the result and make the prediction considering only those features. All input variables are categorical. Firstly, they were encoded with LabelEncoder from sklearn library. Ratings from the dataset have three values: 2, 1, and 0. I splitted restaurants than into three clusters which is described in details in the ‘Methodology’ section. The clusters were marked as nice, good or satisfied.
As I am working with the sample which is relatively small, I decided to predict what category (nice, good or satisfied) a restaurant falls into considering ratings given by users.
Restaurants were divided into groups with KMeans Clustering.
Having the prepared encoded dataset with input variables (restaurant features) and output variables (ratings), I started to apply backward elimination and feature selection to select top features, which mostly affect the result.

For making the prediction I used Amazon SageMaker LinearLearner with multi-class classifier algorithm and created custom Keras model to be able to compare performance of these models. Multiclass classification is a common ML problem where result of the prediction must be assigning on of the finite set of classes. Difference between multi-labelling and multi-class classification is that by multi-class classification I assume that classes are mutually exclusive, while multi-labelling classification allows assigning more then 1 label to the data point. In my case I have exactly multi-class classification task as I need to assign exactly one rating category to the restaurant. According to Amazon documentation for multi class classifier I can use it if my input dataset has numerical features and I know ahead which categories should I get as an output.
Also, I tried to improve results of prediction by using hyper parameter tuning for multi class classifier by balancing class weights so that linear learner can consider label frequencies in my training set. The weights will be the inverses of the frequencies. I do it because after loss function linear learner algorithm optimizes class weights. Class weights put more weight on rarer classes so that the importance of each class is equal. Without class weights, each example in the training set is treated equally. And I want to see what will be the result if categories that occur more frequently in training set have less weight then categories that are rarer.
For another solution I created the custom model from Keras framework and added some basic layers, such as Dense and Activation.
Dense layer will be used for receiving input shape and producing output shape. As activation function Relu was chosen because it was proven to work well in neural networks.
I also added Dropout layer as it is recommended as a technique used to prevent a model from overfitting. Dropout works by randomly setting the outgoing edges of hidden units (neurons that make up hidden layers) to 0 at each update of the training phase. I chose value 0.5 for Dropout layer as I want to exclude random nodes from each update cycle which to improve generalisation and is less likely to overfit the training data. As optimizer I set Adam optimizer as it is the most common choice for optimizer for many use cases.
As loss function I applied categorical cross entropy function as it is generally good for classification problem. A lower score indicates that the model is performing better. For measurement of performance I chose accuracy, so that while training the model I can see the accuracy score on the validation. So, later after applying these two approaches I can compare performance of both of them.

Benchmark

As benchmark for this problem I compared performances of predictions from the Amazon SageMaker multi-class classifier, Sklearn RandomForest classifier and custom Keras model. I will compare all the results and see what model fits better the current project.

III. Methodology

Data Preprocessing

As mentioned, during the exploration of the present data, I found that I am dealing with categorical data. In addition to this, these data were spread across multiple csv files, so data

frames vary in size, shape and indices. For example, there is information about general features of restaurant but there is no information about the rating this particular restaurant has, or a restaurant has multiple ratings given from the different users or a restaurant has duplicated placeID (which is identifier of restaurant) in restaurant_payments.csv if restaurant accepts multiple types of payments. To make processing data possible I made cleaning the data:
- lowercased values in categorical features
- replaced some values that are synonymous to each other by one value, for ex.

For encoding categorical variables I used LabelEncoder, OneHotEncoder and ColumnTransformer. For the last two methods that I used in backward elimination feature selection I also need to remove one of the column corresponding to value of categorical feature to avoid ‘dummy variable trap’.

Implementation

1. K-Means Clustering

For this project I found data about ratings given by users to restaurants. I noticed that all these restaurants are located in Mexico, so I assume the data is probably a part of some bigger data sets and the size of sample data that I will be working with is relatively small. Thus, I set my target in this project not to predict exact rating for restaurant but to predict which category this specific restaurant belongs to.
First of all, I need to define how many and the exact categories I will assign restaurants to. For this task I used unsupervised Machine Learning algorithm which is K-Means clustering basing on mean ratings of restaurants: general, food and service ratings.

Thus, my first step was to group restaurants by their ids and calculate mean values in each type of their ratings. Data exploration told me that from 1161 restaurant ratings I have 130 unique restaurants if I group them by id, so it means that at least one restaurant has multiple ratings and I can calculate mean for it

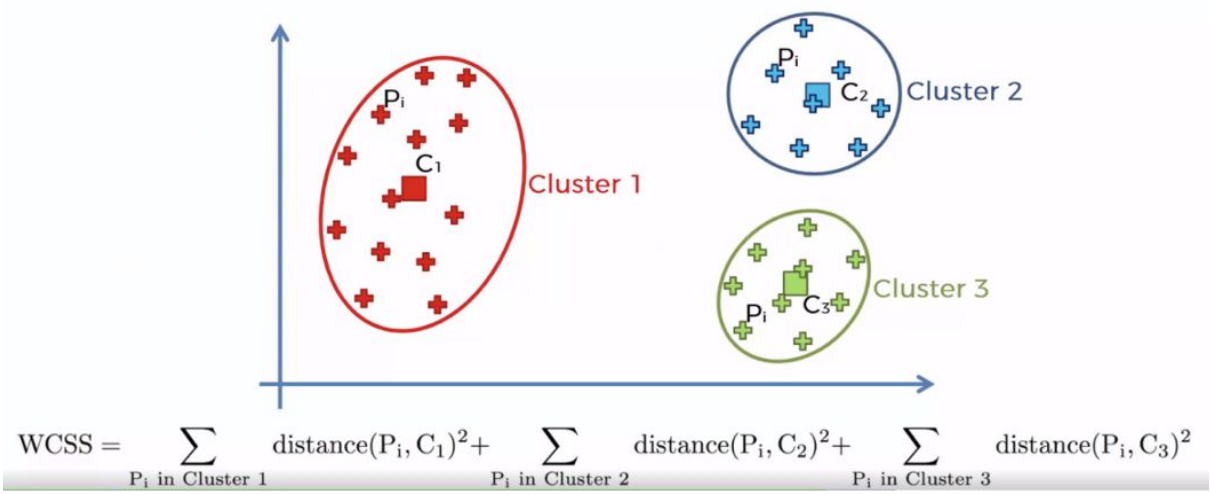
	userID	placeID	rating	food_rating	service_rating
0	U1077	135085	2	2	2
1	U1077	135038	2	2	1
2	U1077	132825	2	2	2
3	U1077	135060	1	2	2
4	U1068	135104	1	1	2

	placeID	rating	service_rating	food_rating
0	132560	0.50000	0.25000	1.00000
1	132561	0.75000	1.00000	1.00000
2	132564	1.25000	1.50000	1.25000
3	132572	1.00000	0.93333	1.00000
4	132583	1.00000	1.25000	1.00000
5	132584	1.33333	1.00000	1.50000

Then basing on those 3 inputs features I found what is the optimal number of clusters (categories of restaurants) for splitting should be chosen. To find the k number of clusters I use Elbow method. This is an empirical way to define more optimal k. The idea is to plot value of cost function which depends on different values of k. And while k grows up then average distortion will go down as centroids get to be closer to centroids of cluster. And by increasing k number, improvements of cost function will decline rapidly in some point, creating the elbow shape. That point is the optimal value for number of clusters. I used here Elbow method with within cluster sum of error (WCSS). WCSS is equal to sum of all distances within clusters between each point of specific cluster and its centroid.

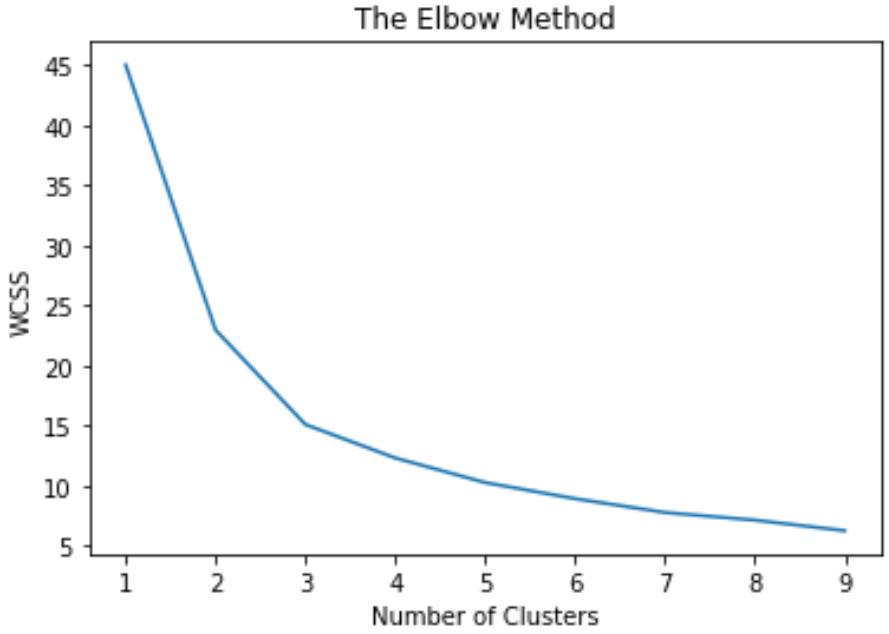
$$WCSS = \sum_{k=1}^{n} (x_i - c_i)^2, \; x_i - \text{point of i-cluster}, c_i - \text{centroid of i-cluster}$$

So for example in case of three clusters calculation of WCSS will look as the following:



From the plot of WCSS built for varying number of clusters I concluded that at the point 3 the graph gets started do decline and creates so-called elbow, so it means that 3 is the optimal number of clusters for restaurants' ratings.

The plot built on my data set



After finding optimal number of restaurants categories, which is 3, I splitted all restaurants into clusters using Sagemaker KMeans estimator. For convenience, I labeled these categories as “Nice”, “Good” and “Satisfied”. As I have 130 unique restaurants from restaurant_ratings dataset, then I got 130 cluster predictions for each of them. If I pick randomly any of them, I can easily understand which category this restaurant belongs to and how far its data point is from its nearest cluster centroid.

```
restaurant_id = 11

print('Restaurant is: ', restaurant_ratings.placeID[restaurant_id])
print(cluster_predictions[restaurant_id])
```

```
Restaurant is:  132630
label {
  key: "closest_cluster"
  value {
    float32_tensor {
      values: 2.0
    }
  }
}
label {
  key: "distance_to_cluster"
  value {
    float32_tensor {
      values: 0.13399739563465118
    }
  }
}
```

After clustering restaurants, I counted how restaurants are distributed among categories and calculated centroids of each cluster:

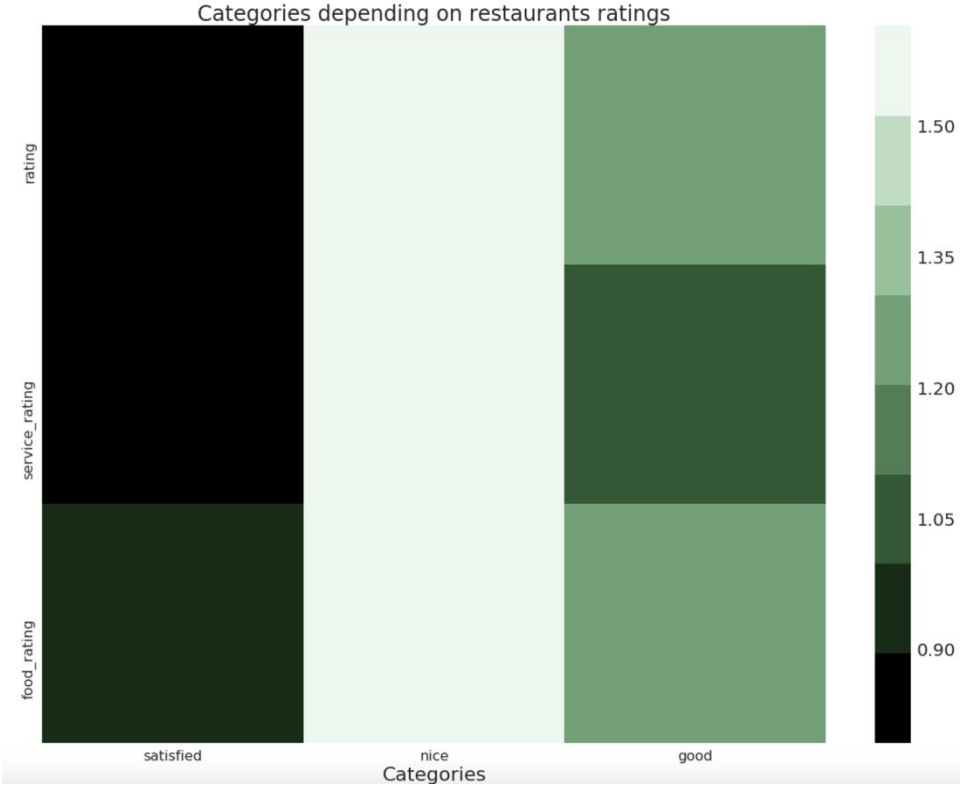
```
# count up the points in each cluster
cluster_df = pd.DataFrame(cluster_prediction_labels)[0].value_counts()

print(cluster_df)
```

```
2.0    63
0.0    39
1.0    28
Name: 0, dtype: int64
```

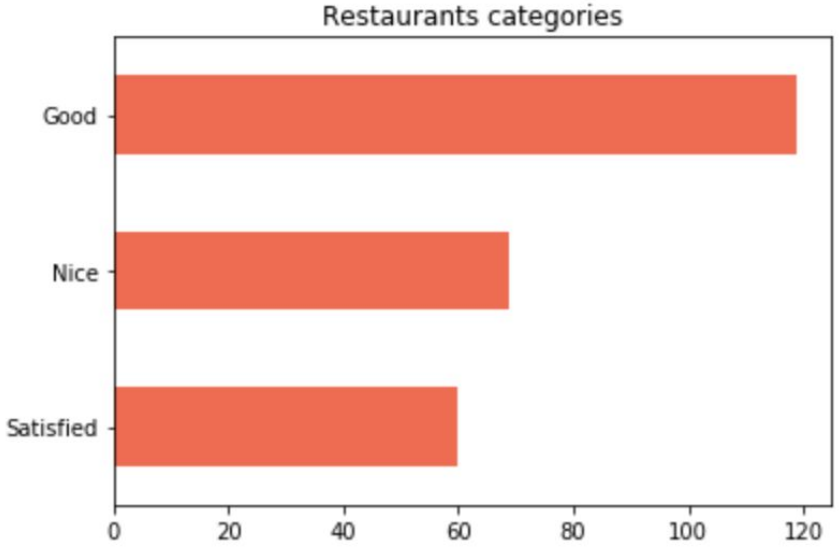
From what I see here, generally, restaurants in cluster #1 have the best ratings, so I label this cluster as “Nice”, the 2nd cluster will have label “Good” (means that restaurants from this category have intermediate ratings), and the last cluster will have “Satisfied” category as its ratings are relatively low. And it is quite reasonable that the biggest cluster is with label “Good” and less restaurants belong to "Nice" and "Satisfied" categories. This is expected distribution, close to Guassian one: here I see distribution of restaurants categories follows central limit theorem: no matter what is the distribution of the population, the shape of the sampling distribution will converge to normal distribution while as the sample size (N) increases and each separate data point impact on the common population is relatively small.

	rating	service_rating	food_rating
0	0.808347	0.792421	0.910435
1	1.614644	1.535016	1.572905
2	1.222862	1.089601	1.248938

	rating	service_rating	food_rating
satisfied	0.808347	0.792421	0.910435
nice	1.614644	1.535016	1.572905
good	1.222862	1.089601	1.248938



Then I created the data frame with restaurants and assigned to each of them its cluster. So my task will be to predict category of restaurant using these pairs of restaurant-category as label to be predicted.



	placeID	rating	service_rating	food_rating	category
0	132560	0.50	0.250000	1.00	0.0
1	132561	0.75	1.000000	1.00	0.0
2	132564	1.25	1.500000	1.25	2.0
3	132572	1.00	0.933333	1.00	0.0
4	132583	1.00	1.250000	1.00	2.0

2. Feature selection

As I use categorical input variables, I declined my preliminary idea to use PCA for reducing input components, because PCA is generally designed to work well with continuous numeric variables. PCA aims to reduce number of principal components by minimizing variance (squared deviations) captured by them but not less, then some defined coverage percentage. The concept of squared deviations breaks down when I have binary variables which is result of encoding categorical features.

2.1 Backward elimination

The idea of backward elimination method is to create optimal matrix of features that contains only those independent variables that are significant for the rating. I use the following algorithm for this:

- Set some significance level (good practice is to have 5%), so my SL will be 0.05
- Fit full model X with all possible features (I used here OLS - ordinary least squares method)
- Look for the feature with highest p-value
- If p-value of variable is higher SL then this independent variable should be removed from model, otherwise it stays. The lower p-value the higher significance has independent variable with respect to dependent variable (restaurant category),

because small p-value means that this feature is unlikely under hypothesis that it does not impact the restaurant rating category.

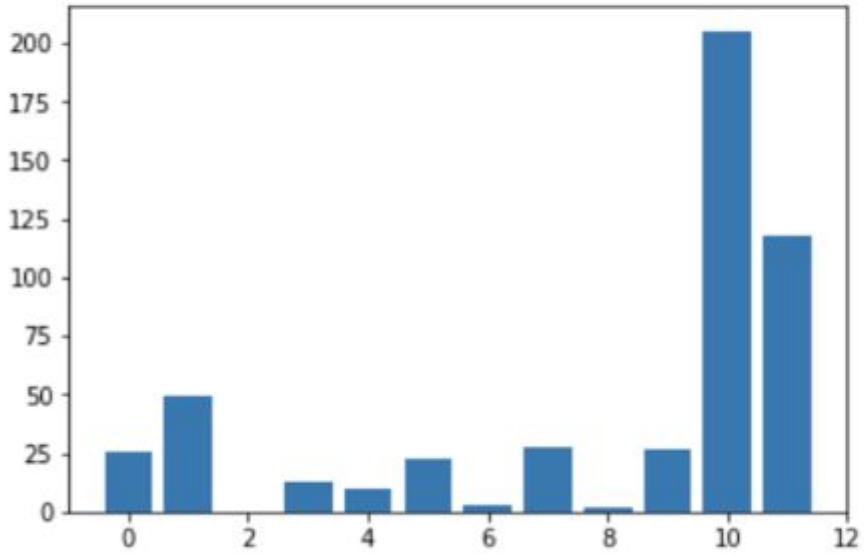
- Fit a model again with left features and repeat step 3 unless there is no possible feature to be removed. After this I should have input data with only relevant features.

From this feature selection I see that actually middle price, Rambience, area, franchise and other services of restaurant have impact on his category. I can use them to predict multi-class classification.

2.2 Feature Selection with chi2 method

Apart from the backward elimination method, I use chi2 test to whether the output is affected by an input variable or it's independent of that input. If the input variable does not make a difference to the result, it will be removed from the data thereby improving the prediction.
The Pearson's Chi-Squared test helps to measure the statistical significance of differences between two or more indicators. To understand the intuition behind the Chi-Squared, I read the article by Jason Brownlee (https://machinelearningmastery.com/chi-squared-test-for-machine-learning/) and other sources he referred in the end.
If the result of the test shows the categorical input variables are independent of the target variable, those variables will be removed and not considered during the prediction.
I used chi2() function from scikit-learn library along with SelectKBest class to select top most relevant features. I defined the SelectKBest class and then applied it to the train set.
The SelectKbest class displays the scores of each variable. The higher score is, the better variable is, i.e. it should be kept in the dataset. I also created a bar chart of variables with scores to have a clear pictures how many features should be selected.

```
visualise_selected_features(chi2_feature_selection)
```



Selected features that are relevant for restaurant rating category:

```
: accuracies[2]

: (94.3820224719101,
   ['alcohol',
    'smoking_area',
    'accessibility',
    'price',
    'Rambience',
    'franchise',
    'area',
    'Rpayment',
    'parking_lot',
    'Rcuisine'])
```

This shows which categories left after excluding non-relevant features:

```
restaurants_with_categories.head()
```

	alcohol	smoking_area	accessibility	price	Rambience	franchise	area	Rpayment	parking_lot	Rcuisine	category
0	no_alcohol_served	not permitted	no_accessibility	medium	familiar	f	closed	cash	public	fast_food	2.0
1	no_alcohol_served	not permitted	completely	low	familiar	f	open	cash	none	mexican	2.0
2	no_alcohol_served	not permitted	no_accessibility	medium	familiar	f	closed	cash	none	seafood	2.0
3	full_bar	not permitted	completely	medium	familiar	t	closed	cash	yes	mexican	2.0
4	full_bar	not permitted	completely	medium	familiar	t	closed	visa	yes	mexican	2.0

Refinement

While looking for appropriate way to reduce number of principal components, I found that there is way kind of PCA to do it. So, I was thinking to apply the PCA for categorical data which is called CATPCA. Categorical PCA also reduces a large number of variables to a smaller number of components. Each component is an uncorrelated combination of the original variables. In order to deal with non-numerical data CATPCA converts categories into numeric values through optimal scaling. The transformed variables (quantified) reflect the distance between different levels of ordinal variables or different categories of nominal variables that optimise the properties of the correlation matrix of the quantified variables.

As I have data for users, I could improve my model to predict category of restaurant based on preferences of particular users. For this I need to make more investigations to find what is the correlation between data in user profile and restaurant features and how it impacts the rating. So, for example, I guess that user who is marked as smoker in his profile, is supposed to give higher rating to restaurant that permits smoking area.

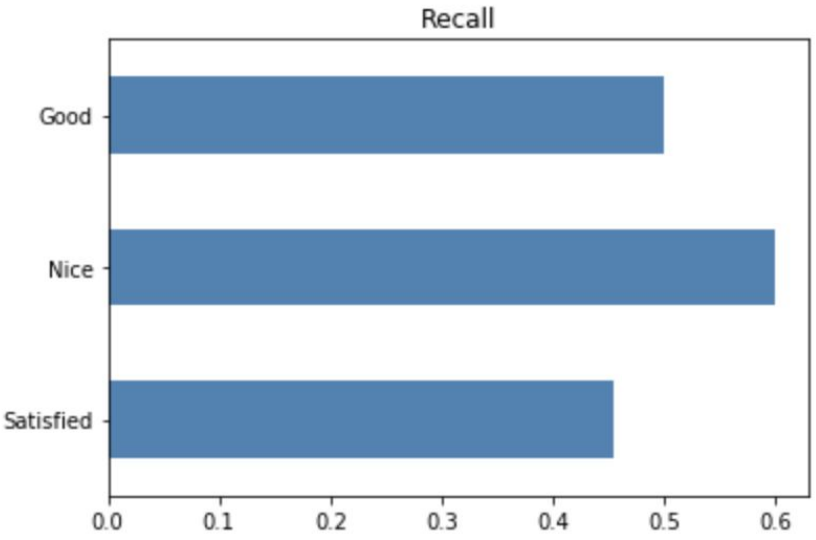
Also, the more advanced task could be create the model that can predict not only category of restaurant rating but also exact value of this rating or rating which has not integer value as

in input data but some continuous number to give more precise information to user how well this restaurant could suits to him.

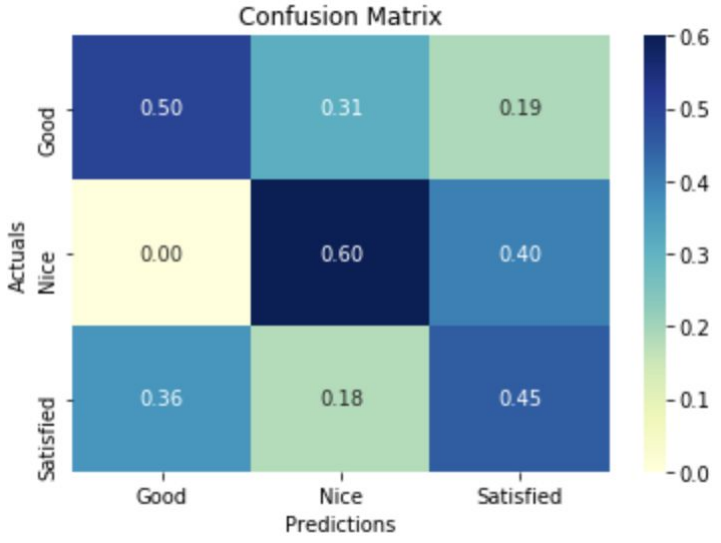
IV. Results

Model Evaluation and Validation

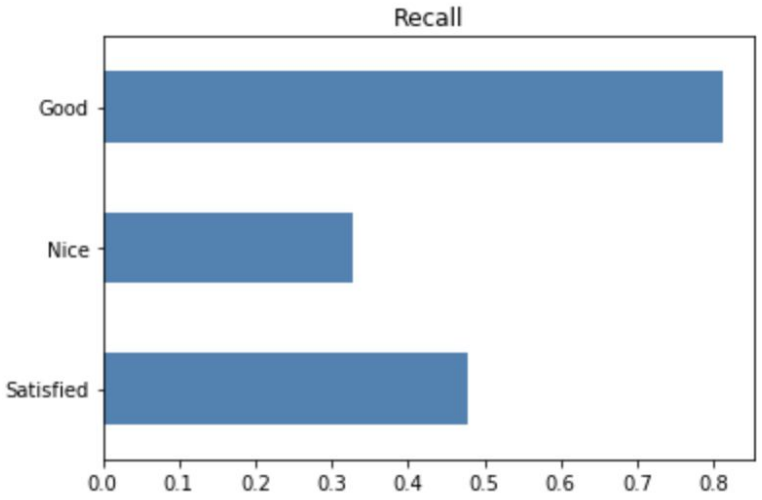
The first evaluation of the model is when I tried to predict restaurant category using all present categorical features. So, with considering all features as relevant I have 50% of properly predicted restaurant categories.

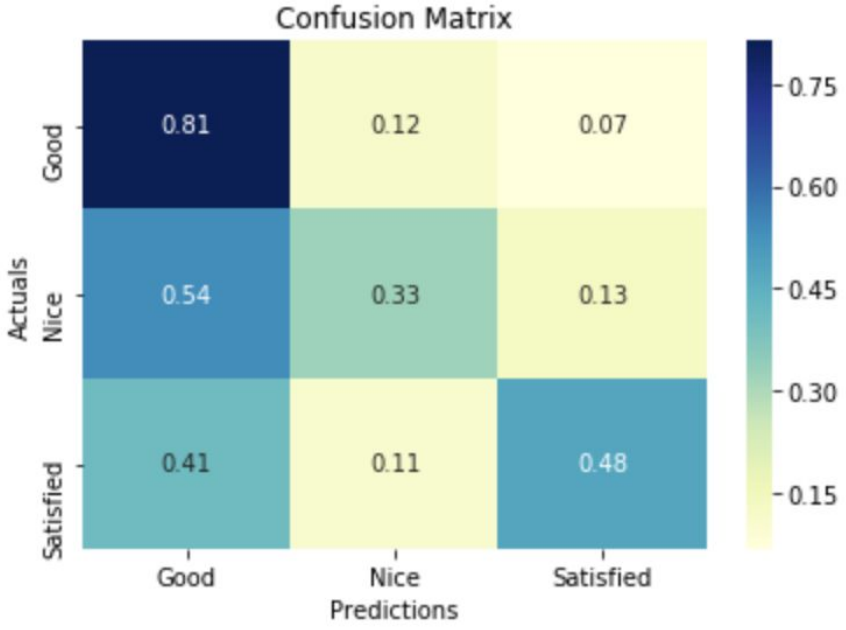


Recall metrics shows the accuracy when true prediction matches the specific category. On the plot above the average of all recall values is shown, this measurement is also called macro recall. The recall achieved by model is different for different categories of restaurants. So, it is higher for 'Nice' category and relatively small for the rest categories.



A confusion matrix is a tool for visualizing the performance of a multiclass model. It has entries for all possible combinations of correct and incorrect predictions, and shows how often each one was made by our model. It has been row-normalized: each row sums to one, so that entries along the diagonal correspond to recall. So, at the first row I see that if restaurant's category is actually 'Good' then my model recognizes it for half of cases. In 31% and 19% of cases it predicts wrong that restaurant has 'Nice' and 'Bad' categories respectively. The accuracy is relatively low because 50% of accuracy means that only in the half of all cases the model can correctly predict restaurant rating category.



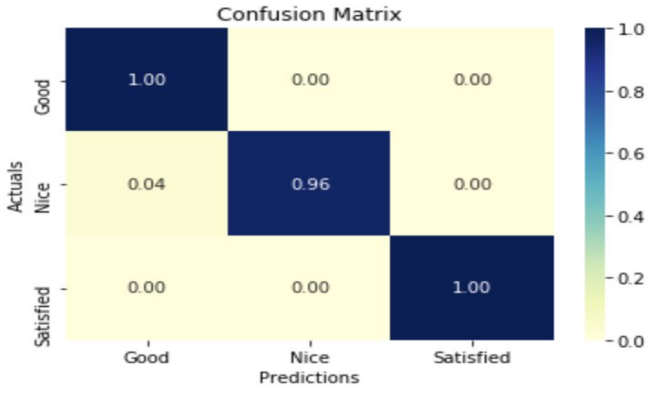


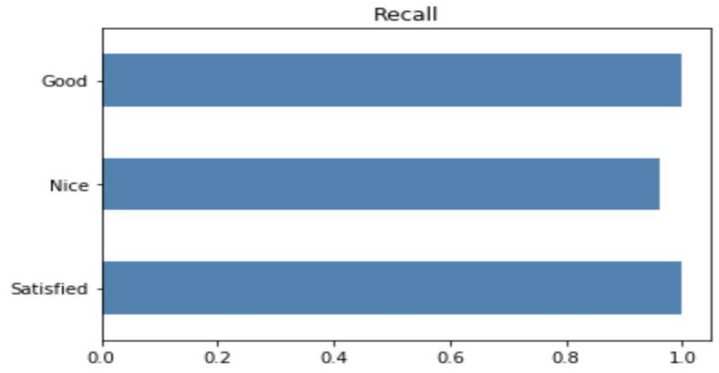
The second evaluation was performed after applying feature selection with chi2 method. The accuracy was significantly increased by 12,5 % when I used 10 more relevant features that have the highest score calculated with chi2 method. The recall achieved by model increased for 'Good' category which is expected as 'Good' category is the most common one.

A confusion matrix was also improved as for now my model predicts right 'Good' category in 81% cases when restaurants have actually 'Good' category. Prediction of 'Satisfied' category also increased a little bit by 3%. However true predictions of 'Nice' category went significantly down.
Results predicted after hyper parameters tuning of multi-class classifier by training with balanced class weights were not improved at all and even decreased by approximately 6%. There is slight increase of recall metrics for 'Satisfied' category of restaurants. The rate of right predicted category in this case grew from 48% to 72%. There is also a small improvement for predicting restaurants with 'Nice' category but the general accuracy decreased by 2%. Also, the common recall metrics increase only for 'Satisfied' category which is not so good as restaurants in this category are less common then the rest so this model does not make good predictions for majority of restaurants.
At the end I used found relevant features from backward. According to this method I got following relevant features:

- 'price_medium'
- 'Rambience_familiar', 'Rambience_quiet'
- 'franchise_f', 'franchise_t'
- 'area_closed', 'area_open',
- 'other_services_internet', 'other_services_none', 'other_services_variety'.

They are designating specific values in categorical features, not the features themselves, but analysing them, I can conclude (that's why I grouped them above) that such categorical features as 'price','Rambience','franchise','area','other_services' have bigger impact on the rating category of restaurant. Thus, I chose them as selected categories, encoded them with LabelEncoder as in last approach and passed to multi-class classifier.





Justification

I applied all methods that were described in the previous sections to get a better results. After the first training with all features considered as relevant I have the following results:
Sklearn RandomForest classifier: accuracy: 94.38 %
Amazon SageMaker multi-class classifier: 53.5 %
Custom model: 69%.
However after removing non-relevant features for restaurants ratings with backward elimination, the performance of Amazon SageMaker multi-class classifier was equal to 99.6%.
After evaluation metrics of my model I see quite essential difference and big improvement in prediction categories. From the confusion matrix, I see that now the model has only small inaccuracy by predicting category of restaurants in 'Nice' category. So, in 4% of cases when restaurant has actually 'Nice' category, model predicts that it belongs to 'Good' category. But generally, the accuracy metrics increased as well recall metrics was improved.