1.0 Introduction

Bengaluru has a fascinating food culture. Restaurants from all over the world can be found here in Bengaluru. From United States to Japan, Russia to Antarctica, you get all type of cuisines here. Delivery, Dine-out, Pubs, Bars, Drinks, Buffet, Desserts, you name it, and Bengaluru has it. Bengaluru is best place for foodies. The number of restaurants is increasing day by day. Currently which stands at approximately 12,000 restaurants. With such a high number of restaurants. This industry hasn't been saturated yet. And new restaurants are opening every day. However, it has become difficult for them to compete with already established restaurants. The key issues that continue to pose a challenge to them include high real estate costs, rising food costs, shortage of quality manpower, fragmented supply chain and over-licensing.

Bengaluru is India's IT capital. Most people here are mostly reliant on restaurant food because they don't have time to prepare for themselves. It has thus become important to research the demographics of a place with such an enormous demand for restaurants. What kind of food in a town is more prevalent? The whole town enjoys vegetarian food. If yes, then that place is inhabited, for example, by a specific sect of people. Jain, Marwaris, mainly vegetarian Gujaratis. Using the data, this form of analysis can be done by analysing various factors. . Bangalore (officially known as Bengaluru) is the capital and largest city of the Indian state of Karnataka. With a population of over 15 million, Bangalore is the third largest city in India and 27th largest city in the world. Bangalore is one of the most ethnically diverse cities in the country, with over 51% of the city's population being migrants from other parts of India. Bangalore is sometimes referred to as the "Silicon Valley of India" because of its role as the nation's leading information technology (IT) exporter.

Bangalore has a unique food culture. Restaurants from all over the world can be found here in Bengaluru, with various kinds of cuisines. Some might even say that Bangalore is the best place for foodies. The growing number of restaurants and dishes in Bangalore is what attracts me to inspect the data to get some insights, some interesting facts and figures. So, in this article I will be analysing the Zomato restaurant data for the city.

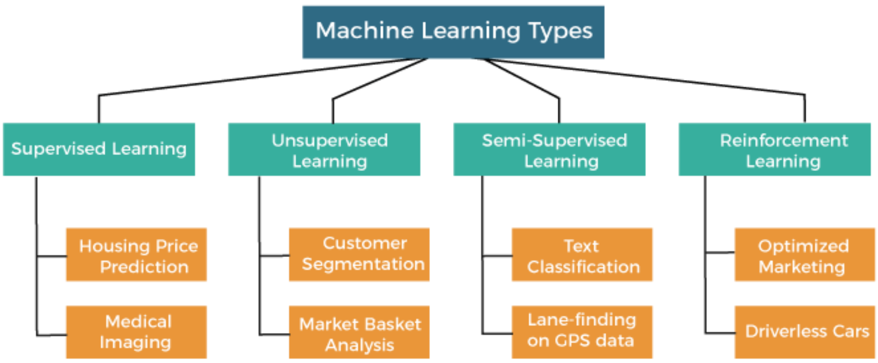
1.1. What are the different types of Machine Learning?

Machine learning is a subset of AI, which enables the machine to automatically learn from data, improve performance from past experiences, and make predictions. Machine learning contains a set of algorithms that work on a huge amount of data. Data is fed to these algorithms to train them, and based on training, they build the model & perform a specific task.

These ML algorithms help to solve different business problems like Regression, Classification, Forecasting, Clustering, and Associations, etc.

Based on the methods and way of learning, machine learning is divided into mainly four types, which are:

1. Supervised Machine Learning
2. Unsupervised Machine Learning
3. Semi-Supervised Machine Learning
4. Reinforcement Learning



Supervised Learning:

Supervised learning is one of the most basic types of machine learning. In this type, the machine learning algorithm is trained on labelled data. Even though the data needs to be labelled accurately for this method to work, supervised learning is extremely powerful when used in the right circumstances.

In supervised learning, the ML algorithm is given a small training dataset to work with. This training dataset is a smaller part of the bigger dataset and serves to give the algorithm a basic idea of the problem, solution, and data points to be dealt with. The training dataset is also very similar to the final dataset in its characteristics and provides the algorithm with the labelled parameters required for the problem.

The algorithm then finds relationships between the parameters given, essentially establishing a cause-and-effect relationship between the variables in the dataset. At the end of the training, the algorithm has an idea of how the data works and the relationship between the input and the output.

This solution is then deployed for use with the final dataset, which it learns from in the same way as the training dataset. This means that supervised machine learning algorithms will continue to improve even after being deployed, discovering new patterns and relationships as it trains itself on new data.

Categories of Supervised Machine Learning

Supervised machine learning can be classified into two types of problems, which are given below:

- Classification
- Regression

Classification

Classification algorithms are used to solve the classification problems in which the output
variable is categorical, such as Yes; or No, Male or Female, Red or Blue, etc. The classification algorithms predict the categories present in the dataset. Some real-world examples of classification algorithms are Spam Detection, Email filtering, etc.

Some popular classification algorithms are given below:

- Random Forest Algorithm
- Decision Tree Algorithm
- Logistic Regression Algorithm
- Support Vector Machine Algorithm

Regression

Regression algorithms are used to solve regression problems in which there is a linear relationship between input and output variables. These are used to predict continuous output variables, such as market trends, weather prediction, etc.

Some popular Regression algorithms are given below:

- Simple Linear Regression Algorithm
- Multivariate Regression Algorithm
- Decision Tree Algorithm
- Lasso Regression

Applications of Supervised Learning:

- Image Segmentation
- Medical Diagnosis
- Fraud Detection
- Spam detection
- Speech Recognition

Unsupervised Learning:

Unsupervised machine learning holds the advantage of being able to work with unlabelled data. This means that human labour is not required to make the dataset machine-readable, allowing much larger datasets to be worked on by the program.

In supervised learning, the labels allow the algorithm to find the exact nature of the relationship between any two data points. However, unsupervised learning does not have labels to work off, resulting in the creation of hidden structures. Relationships between data points are perceived by the algorithm in an abstract manner, with no input required from human beings.

The creation of these hidden structures is what makes unsupervised learning algorithms versatile. Instead of a defined and set problem statement, unsupervised learning algorithms can adapt to the data by dynamically changing hidden structures. This offers more post-deployment development than supervised learning algorithms.

Categories of Unsupervised Machine Learning:

Unsupervised Learning can be further classified into two types, which are given below:

- Clustering
- Association

Clustering

The clustering technique is used when we want to find the inherent groups from the data. It is a way to group the objects into a cluster such that the objects with the most similarities remain in one group and have fewer or no similarities with the objects of other groups. An example of the clustering algorithm is grouping the customers by their purchasing behaviour.
Some of the popular clustering algorithms are given below:

- K-Means Clustering algorithm
- Mean-shift algorithm
- DBSCAN Algorithm
- Principal Component Analysis
- Independent Component Analysis

Association

Association rule learning is an unsupervised learning technique, which finds interesting relations among variables within a large dataset. The main aim of this learning algorithm is to

find the dependency of one data item on another data item and map those variables accordingly so that it can generate maximum profit. This algorithm is mainly applied in Market Basket analysis, Web usage mining, continuous production etc.
Some popular algorithms of Association rule learning are:

- Apriorism Algorithm
- Eclat
- FP-growth algorithm.

Applications of Unsupervised Learning:

- Network Analysis
- Recommendation Systems.
- Anomaly Detection
- Singular Value Decomposition

1.2. Benefits of Using Machine Learning in Restaurant Success Prediction

Below listed are various uses of AI in Food Sector

1 Start from food market analysis

2 Cleaning equipment that does not need disassembling (CIP)

3 Better hygiene – Kankan AI in the Food and Beverage industry solution.

4 Food and beverage supply chain optimization

5 Using AI in Food Industry: Machine Learning applications in Food Manufacturing

5.1 Supply chain optimization – less waste and more transparency

5.2 Sorting food: optical sorting solutions

5.3 Predictive maintenance, remote monitoring and condition monitoring

6 Machine learning applications in the restaurant business

6.1 Analytical solutions for a better customer experience

6.2 Food-selling site and applications

6.3 AIs for online restaurant search

6.4 Voice searches

6.5 Self-serving system

6.6 Innovations in robotics for the food industry

AI in Food and Beverage Statistics

1.3. About Online Food Industry

Machine learning supports the manufacturing process and the restaurant business at every stage, decreasing expenses and improving quality. Artificial Intelligence and Machine Learning solutions offer large possibilities to optimize and automate processes, save costs and make less human error possible for many industries. Food and Beverage is not an exception, where it can be beneficially applied in restaurants, bar and cafe businesses as well as in food manufacturing. These two segments have common use cases where AI in the food industry can be applied, as well as different ones, what is linked to different problems that must be solved.

1.3.1 AI / ML Role in Food Industry

Machine Learning is a sub-set of artificial intelligence where computer algorithms are used to autonomously learn from data. Machine learning (ML) is getting more and more attention and is becoming increasingly popular in many other industries. The food-based industry is stuffed with many well-established brands as well as food outlets. Due to the growing competition, this industry is losing its attraction for establishing a new business. In the food industry, using technology, especially data science, is the only way which can make anyone stay upfront in the competition.

2.0. Zomato Bangalore Restaurants Success Prediction

This Zomato data aims at analysing demography of the location. Most importantly it will help new restaurants in deciding their theme, menus, cuisine, cost etc for a particular location. It also aims at finding similarity between neighbourhoods of Bengaluru based on food. The dataset also contains reviews for each of the restaurant which will help in finding overall rating for the place.

The basic idea of analysing the Zomato dataset is to get a fair idea about the factors affecting the establishment of different types of restaurants at different places in Bengaluru, aggregate rating of each restaurant, Bengaluru being one such city has more than 12,000 restaurants with restaurants serving dishes from all over the world. With each day new restaurants opening the industry hasn't been saturated yet and the demand is increasing day by day. Inspire of increasing demand it however has become difficult for new restaurants to compete with established restaurants. Most of them serving the same food. Bengaluru being an IT capital of India. Most of the people here are dependent mainly on the restaurant food as they don't have time to cook for themselves.

With such an overwhelming demand of restaurants it has therefore become important to study the demography of a location. What kind of a food is more popular in a locality? Do the entire locality loves vegetarian food. If yes then is that locality populated by a particular sect of people for e.g., Jain, Marwaris, Gujaratis who are mostly vegetarian. This kind of analysis can be done using the data, by studying the factors such as

- Location of the restaurant
- Approx. Price of food
- Theme based restaurant or not
- Which locality of that city serves those cuisines with maximum number of restaurants
- The needs of people who are striving to get the best cuisine of the neighbourhood
- Is a particular neighbourhood famous for its own kind of food.

"Just so that you have a good meal the next time you step out"

2.1 Main Drivers for Zomato Restaurant Success Prediction

Predictive modelling allows for simultaneous consideration of many variables and quantification of their overall effect. When many restaurants are analysed, the factors contributing to the success begin to emerge.

The following are the main drivers which influencing the Restaurant Success Prediction:

• Food. • Consistency. • Service. • Environment. • Reputation. • Convenience. • Value. • Innovation	• User-friendly interfaces • Fast Delivery • Variety of Restaurant availability • Easy Payment modes • Attention to customers queries and feedback

2.2. Internship Project - Data Link

The internship project data has taken from Kaggle, and the link is:

https://www.kaggle.com/datasets/himanshupoddar/zomato-bangalore-restaurants

This dataset contains 17 columns and 50000 rows approximately.

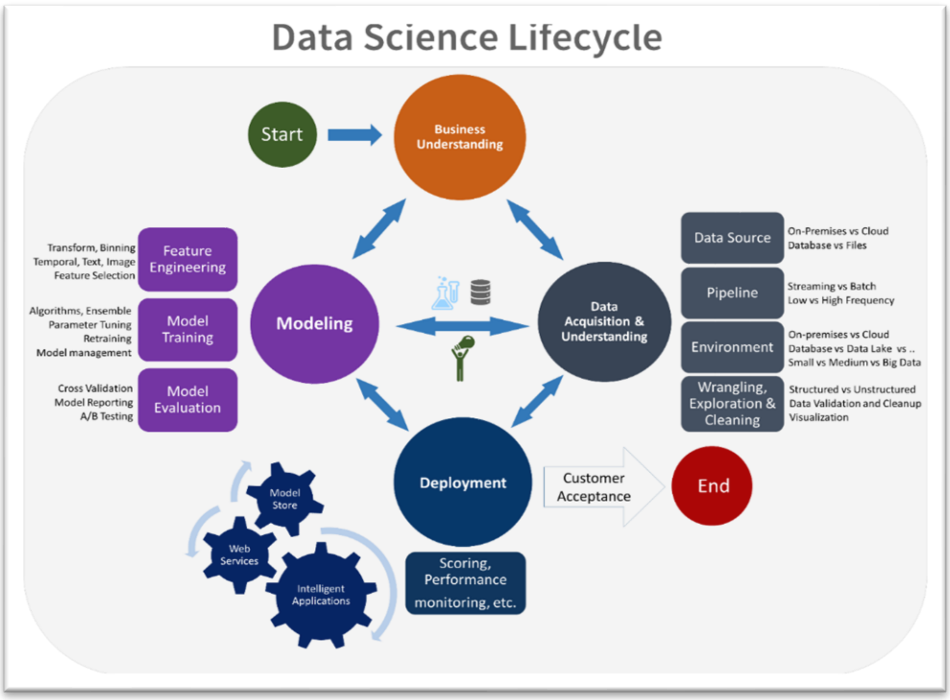
3.0 AI / ML Modelling and Results

3.1. Your Problem of Statement

Zomato is an Online food ordering service, serving worldwide in which users can order food from the website or from mobile based applications. For this business problem, we are restricting only to the Bangalore region and Bangalore based restaurants. Dataset was created by extracting (web scraping) the information such as Approx. Price of food, Theme based restaurant or not, aggregate rating of each restaurant etc. about the existing established restaurants serving through Zomato and made available on Kaggle March 2019. The basic idea of analyzing the Zomato dataset is to get a fair idea about the factors affecting the establishment of different types of the restaurant at different places in Bangalore. While establishing a new restaurant some of the common questions such as • Location of the restaurant • Approx. Price of food etc. should be wisely answered to be successful in this industry. For the given historical data, to predict whether a new restaurant can be successful or not (positive or negative).

3.2. Data Science Project Life Cycle

Data Science is a multidisciplinary field of study that combines programming skills, domain expertise and knowledge of statistics and mathematics to extract useful insights and knowledge from data.



3.2.1. Data Exploratory Analysis

Exploratory data analysis (*EDA*) is an especially important activity. It enables an in depth understanding of the dataset, define or discard hypotheses and create predictive models on a solid basis. It uses data manipulation techniques and several statistical tools to describe and understand the relationship between variables and how these can impact business.

3.2.2. Data Pre-processing

We removed variables which does not affect our target variable (RATING) as they may add noise and increase our computation time, we checked the data for anomalous data points and outliers. We did principal component analysis on the data set to filter out unnecessary variables and to select only the important variables which have greater correlation with our target variable.

3.2.2.1. Check the Duplicate and low variation data

An important part of Data analysis is analyzing Duplicate Values and removing them. duplicated () method helps in analyzing duplicate values only. It returns a Boolean series which is True only for Unique elements.

3.2.2.2. Identify and address the missing variables

We can check for null values in a derived dataset. But, sometimes, it might not be this simple to identify missing values. One needs to use the domain knowledge and look at the data description to understand the variables. There are variables that have a minimum value of zero. On some columns, a value of zero does not make sense and indicates an invalid or missing value.

Quick Classification of Missing Data

There are three types of missing data as below:

Missing Completely at Random (MCAR): It is the highest level of randomness. This means that the missing values in any features are not dependent on any other feature's values. This is the desirable scenario in case of missing data.

Missing At Random (MAR): This means that the missing values in any feature are dependent on the values of other features.

Missing Not at Random (MNAR): Missing not at random data is a more serious issue and, in this case, it might be wise to check the data gathering process further and try to understand why the information is missing.

What to Do with the Missing Values?

We identified the missing values in a derived dataset, next we should decide the further course of action.

The missing values:

- Missing **Ignore** data under 10% for an individual case or observation can generally be ignored, except when the missing data is a MAR or MNAR.
- The number of complete cases i.e., observation with no missing data must be sufficient for the selected analysis technique if the incomplete cases are not considered.

Drop the missing values:

- Dropping a variable
- If the data is MCAR or MAR and the number of missing values in a feature is very high, then that feature should be left out of the analysis.
- If missing data for a certain feature or sample is more than 5% then you probably should leave that feature or sample out.
- If the cases or observations have missing values for target variables(s), it is advisable to delete the dependent variable(s) to avoid any artificial increase in relationships with independent variables.

Case Deletion:

In this method, cases which have missing values for one or more features are deleted. If the cases having missing values are small, it is better to drop them. Though this is an easy approach, it might lead to a significant decrease in the sample size. Also, the data may not always be missing completely at random. This may lead to biased estimation of parameters.

Imputation:

Imputation is the process of substituting the missing data by some statistical methods. Imputation is useful in the sense that it preserves all cases by replacing missing data with an estimated value based on other available information. But imputation methods should be used carefully as most of them introduce a large amount of bias and reduce variance in the dataset.

In Pandas missing data is represented by two values:

None: None is a Python singleton object that is often used for missing data in Python code.

Nan: Nan (an acronym for Not a Number), is a special floating-point value recognized by all systems that use the standard IEEE floating-point representation.

Treating missing data:

After classified the patterns in missing values, it needs to treat them.

Deletion:

The Deletion technique deletes the missing values from a dataset. followings are the types of missing data.

Listwise deletion:

Listwise deletion is preferred when there is a Missing Completely at Random case. In Listwise deletion entire rows (which hold the missing values) are deleted. It is also known as complete-case analysis as it removes all data that have one or more missing values.

In python we use **dropna()** function for Listwise deletion.

Listwise deletion is not preferred if the size of the dataset is small as it removes entire rows if we eliminate rows with missing data then the dataset becomes very short, and the machine learning model will not give good outcomes on a small dataset.

Pairwise Deletion:

Pairwise Deletion is used if missingness is missing completely at random i.e., MCAR.

Pairwise deletion is preferred to reduce the loss that happens in Listwise deletion. It is also called an available-case analysis as it removes only null observation, not the entire row.

Note: All methods in pandas like mean, sum, etc. intrinsically skip missing values.

Dropping complete columns :

If a column holds a lot of missing values, say more than 80%, and the feature is not meaningful, that time we can drop the entire column.

3.2.2.3. Handling of Outliers

What is an Outlier?

An outlier is a data point in a data set that is distant from all other observations.

A data point that lies outside the overall distribution of dataset (95% of customer behavior / claims / spending nature of customer)

What is the reason for an outlier to exist in dataset?

- Variability in the data
- An Experimental measurement errors

How can we Identify an outlier?

- Using Box plots
- Using Scatter plot
- Using Z score

3.2.2.4. Categorical data and Encoding Techniques

Encoding is a technique of converting categorical variables into numerical values so that it could be easily fitted to a machine learning model.

Before getting into the details, let's understand about the different types of categorical variables.

NOMINAL CATEGORICAL VARIABLE:

Nominal categorical variables are those for which we do not have to worry about the arrangement of the categories.

ORDINAL CATEGORICAL VARIABLE:

Ordinal categories are those in which we must worry about the rank. These categories can be rearranged based on ranks.

Now that we have discussed about the type of categorical variables, let's see the different types of encoding:

Nominal Encoding

Ordinal Encoding

1. ONE HOT ENCODING

This method is applied to nominal categorical variables.

To overcome the Disadvantage of Label Encoding as it considers some hierarchy in the columns which can be misleading to nominal features present in the data. we can use the One-Hot Encoding strategy.

One-hot encoding is processed in 2 steps:

Splitting of categories into different columns.

Put ‘0 for others and ‘1’ as an indicator for the appropriate column.

2. Label Encoding:

Label encoding algorithm is quite simple, and it considers an order for encoding, hence can be used for encoding ordinal data.

3. Ordinal Encoding:

We can use Ordinal Encoding provided in Scikit learn class to encode Ordinal features. It ensures that ordinal nature of the variables is sustained.

4. Frequency Encoding:

We can also encode considering the frequency distribution. This method can be effective at times for nominal features.

5. Binary Encoding:

Initially, categories are encoded as Integer and then converted into binary code, then the digits from that binary string are placed into separate columns.

3.2.2.5. Feature Scaling

Feature Scaling is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitudes or values or units.

Normalization

Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

Standardization

Standardization is another scaling technique where the values are centred around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation

3.2.3. Selection of Dependent and Independent variables

The dependent or target variable here is Claimed Target which tells us a particular policy holder has filed a claim or not the target variable is selected based on our business problem and what we are trying to predict.

The independent variables are selected after doing exploratory data analysis and we used Boruta to select which variables are most affecting our target variable.

3.2.4. Data Sampling Methods

The data we have is highly unbalanced data so we used some sampling methods which are used to balance the target variable so we our model will be developed with good accuracy and precision. We used three Sampling methods

3.2.4.1. Stratified sampling

Stratified sampling randomly selects data points from majority class so they will be equal to the data points in the minority class. So, after the sampling both the class will have same no of observations. It can be performed using strata function from the library sampling.

3.2.4.2. Simple random sampling

Simple random sampling is a sampling technique where a set percentage of the data is selected randomly. It is generally done to reduce bias in the dataset which can occur if data is selected manually without randomizing the dataset. We used this method to split the dataset into train dataset which contains 70% of the total data and test dataset with the remaining 30% of the data.

3.2.5. Models Used for Development

We built our predictive models by using the following ten algorithms

3.2.5.1. Model 01 – Logistic Regression

The dependent variable is of binary type (dichotomous) in logistic regression. This type of regression analysis describes data and explains the relationship between one dichotomous variable and one or more independent variables. Logistic regression is used in predictive analysis where pertinent data predict an event probability to a logit function. Thus, it is also called logit regression.

3.2.5.2. Model 02 – Decision Trees

With a decision tree, you can visualize the map of potential results for a series of decisions. It enables companies to compare possible outcomes and then take a straightforward decision based on parameters such as advantages and probabilities that are beneficial to them. Decision tree algorithms can potentially anticipate the best option based on a mathematical construct and come in handy while brainstorming over a specific decision. The tree starts with a root node (decision node) and then branches into sub-nodes representing potential outcomes. Each outcome can further create child nodes that can open other possibilities. The algorithm generates a tree-like structure that is used for classification problems.

3.2.5.3. Model 03 – Random Forest

Random forest algorithms use multiple decision trees to handle classification and regression problems. It is a supervised machine learning algorithm where different decision trees are built on different samples during training. These algorithms help estimate missing data and tend to keep the accuracy intact in situations when a large chunk of data is missing in the dataset. Random forest algorithms follow these steps: Random Forest algorithms follow these steps:

- Select random data samples from a given data set.
- Build a decision tree for each data sample and provide the prediction result for each decision tree.
- Carry out voting for each expected result.
- Select the final prediction result based on the highest voted prediction result.

3.2.5.4. Model 04 – KNN CLASSIFICATION

The *K Nearest Neighbours (KNN) algorithm* is used for both classification and regression problems. It stores all the known use cases and classifies new use cases (or data points) by segregating them into different classes. This classification is accomplished based on the similarity score of the recent use cases to the available ones.

KNN is a *supervised machine learning algorithm,* wherein 'K' refers to the number of neighbouring points we consider while classifying and segregating the known n groups. The algorithm learns at each step and iteration, thereby eliminating the need for any specific learning phase. The classification is based on the neighbour's majority vote.

The algorithm uses these steps to perform the classification:

- For a training dataset, calculate the distance between the data points that are to be classified and the rest of the data points.
- Choose the closest 'K' elements based on the distance or function used.
- Consider a *'majority vote'* between the K points–the class or label dominating all data points reveals the final ranking.

3.2.5.5. Model 05 – Naïve Bayes

Naive Bayes refers to a probabilistic machine learning algorithm based on the Bayesian probability model and is used to address classification problems. The fundamental assumption of the algorithm is that features under consideration are independent of each other and a change in the value of one does not impact the value of the other.

3.2.5.6. Model 06 – Support Vector Machines (SVM)

Support vector machine algorithms are used to accomplish both classification and regression tasks. These are supervised machine learning algorithms that plot each piece of data in the n-dimensional space, with n referring to the number of features. Each feature value is associated with a coordinate value, making it easier to plot the features.

3.2.5.7. Model 07 – Extra Trees

Extra Trees is an ensemble machine learning algorithm that combines the predictions from many decision trees. It is related to the widely used random forest algorithm. It can often achieve as good or better performance than the random forest algorithm, although it uses a simpler algorithm to construct the decision trees used as members of the ensemble.

It is also easy to use given that it has few key hyperparameters and sensible heuristics for configuring these hyperparameters.

3.2.5.8. Model 08 – LGBM Classifier

LightGBM is a gradient boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient with the following advantages:

- Faster training speed and higher efficiency.
- Lower memory usage.
- Better accuracy.
- Support of parallel and GPU learning.
- Capable of handling large-scale data.

3.2.5.9. Model 09 – XGBoost Classifier

XGBoost is an implementation of Gradient Boosted decision trees. XGBoost models majorly dominate in many Kaggle Competitions.

In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.

3.2.5.10. Model 10 – Bagging Classifier

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Bagging reduces overfitting (variance) by averaging or voting, however, this leads to an increase in bias, which is compensated by the reduction in variance though.

3.2.5.11. Model 11 – Gradient Boosting

Gradient Boosting is a popular boosting algorithm. In gradient boosting, each predictor corrects its predecessor's error. In contrast to Adaboost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of predecessor as labels.

There is a technique called the Gradient Boosted Trees whose base learner is CART (Classification and Regression Trees).

3.3. AI / ML Models Analysis and Final Results

We used our train dataset to build the above models and used our test data to check the accuracy and performance of our models. We used confusion matrix to check accuracy, Precision, Recall and F1 score of our models and compare and select the best model for given auto dataset of size ~ 53000 records.

3.3.1. Different Model codes

- The Python code for models are as follows:

```
# To build the 'Multinominal Regression' models with random sampling

from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import ExtraTreesClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive_bayes import GaussianNB

from sklearn.svm import SVC

from sklearn.ensemble import BaggingClassifier

from sklearn.ensemble import GradientBoostingClassifier
```

```
from xgboost import XGBClassifier
import lightgbm as lgb
# Create an object for model
dfLR = LogisticRegression(multi_class='multinomial', penalty='none', solver='newton-cg', random_state=42)
dfDT = DecisionTreeClassifier()
dfRF = RandomForestClassifier()
dfET = ExtraTreesClassifier()
dfKNN = KNeighborsClassifier(n_neighbors=5)
dfGNB = GaussianNB()
dfSVC = SVC(probability=True)
dfBAG = BaggingClassifier(base_estimator=None, n_estimators=100, max_samples=1.0, max_features=1.0,bootstrap=True, bootstrap_features=False, oob_score=False, warm_start=False,
                    n_jobs=None, random_state=None, verbose=0)
dfGB = GradientBoostingClassifier(loss='deviance', learning_rate=0.1, n_estimators=100, subsample=1.0,
                        criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1,
                        min_weight_fraction_leaf=0.0, max_depth=3)
dfXGB = XGBClassifier(n_estimators=100, max_depth=3, eval_metric='mlogloss')
dfLGB = lgb.LGBMClassifier()
# Evalution matrix for all the algorithms
#MM = [ModelLR, ModelDC, ModelRF, ModelET, ModelKNN, ModelGNB, ModelSVM, modelXGB, modelLGB]
MM = [dfLR, dfDT, dfRF, dfET, dfKNN,dfGNB, dfBAG,dfGB,dfSVC , dfXGB, dfLGB]
for models in MM:
```

```
# Train the model with training data
models.fit(x_train,y_train)
# Predict the model with test data set
y_pred = models.predict(x_test)
y_pred_prob = models.predict_proba(x_test)
# Print the model name
print('Model Name: ', models)
# confusion matrix in sklearn
from sklearn.metrics import multilabel_confusion_matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from math import sqrt
print(confusion_matrix(y_pred, y_test)) # Verticle is actual values & horizontal is predicted values
# Actual and predicted classes
lst_actual_class = y_test
lst_predicted_class = y_pred
lst_predicted_prob_class = y_pred_prob
# Class = Label 0-12
lst_classes = [0, 1, 2, 3]
# Compute multi-class confusion matrix
arr_out_matrix = multilabel_confusion_matrix(lst_actual_class, lst_predicted_class, labels=lst_classes)
# Temp store results
model_acc = [];
model_recall = [];
```

```
model_prec = [];
model_fscore = [];
model_spec = [];
model_bal_acc = [];
model_mcc = [];
for no_class in range(len(lst_classes)):
    arr_data = arr_out_matrix[no_class];
    print("Print Class: {0}".format(no_class));
    tp = arr_data[1][1]
    fn = arr_data[0][1]
    tn = arr_data[0][0]
    fp = arr_data[1][0]
    sensitivity = round(tp/(tp+fn), 3);
    specificity = round(tn/(tn+fp), 3);
    accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
    balanced_accuracy = round((sensitivity+specificity)/2, 3);
    precision = round(tp/(tp+fp), 3);
    f1Score = round((2*tp/(2*tp + fp + fn)), 3);
    mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
    MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
    model_acc.append(accuracy);
    model_prec.append(precision);
    model_recall.append(sensitivity);
    model_fscore.append(f1Score);
    model_spec.append(specificity);
```

```
    model_bal_acc.append(balanced_accuracy);
    model_mcc.append(MCC);
    print("TP={0}, FN={1}, TN={2}, FP={3}".format(tp, fn, tn, fp));
    print("Accuracy: {0}".format(accuracy));    # Accuracy score
    print("Precision: {0}".format(precision)); # Precision score
    print("Sensitivity: {0}".format(sensitivity)); # Recall score
    print("F1-Score: {0}".format(f1Score)); # F1 score
    print("Specificity: {0}".format(specificity)); # True Nagative Rate
    print("Balanced Accuracy: {0}".format(balanced_accuracy)); # Balance accuracy score
    print("MCC: {0}\n".format(MCC)); # Matthews Correlation Coefficient
   # OVERALL - FINAL PREDICTION PERFORMANCE
    # importing mean()
from statistics import mean
import math
print("Overall Performance Prediction:");
print("Accuracy: {0}%".format(round(mean(model_acc)*100, 4)));
print("Precision: {0}%".format(round(mean(model_prec)*100, 4)));
print("Recall or Sensitivity: {0}%".format(round(mean(model_recall)*100, 4)));
print("F1-Score: {0}".format(round(mean(model_fscore), 4)));
print("Specificity or True Nagative Rate: {0}%".format(round(mean(model_spec)*100,
4)));
print("Balanced Accuracy: {0}%\n".format(round(mean(model_bal_acc)*100, 4)));
print("MCC: {0}\n".format(round(mean(model_mcc), 4)))
# ROC curve for Multi classes
from sklearn.multiclass import OneVsRestClassifier
```

```
from sklearn.metrics import roc_curve, roc_auc_score
fpr = {}
tpr = {}
thresh ={}
n_class = 4
for i in range(n_class):
    fpr[i], tpr[i], thresh[i] = roc_curve(lst_actual_class, lst_predicted_prob_class[:,i], pos_label=i)
# plotting
plt.plot(fpr[0], tpr[0], linestyle='--',color='brown', label='Class 0 vs Rest')
plt.plot(fpr[1], tpr[1], linestyle='--',color='green', label='Class 1 vs Rest')
plt.plot(fpr[2], tpr[2], linestyle='--',color='blue', label='Class 2 vs Rest')
plt.plot(fpr[3], tpr[3], linestyle='--',color='red', label='Class 3 vs Rest')
plt.title('Multiclass ROC curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive rate')
plt.legend(loc='best')
plt.savefig('Log_ROC')
plt.show()
# ROC AUC score - one-vs-one (OvO) algorithm computes the average of the ROC AUC scores for each class against all other classes
print('roc_auc_score:', round(roc_auc_score(lst_actual_class, lst_predicted_prob_class, multi_class='ovo', average='weighted'),3))
print('--------------------------------------------------------------------------------------------')
new_row = {'Model Name' : models,
           'True_Positive' : tp,
```

```
        'False_Negative' : fn,
        'False_Positive' : fp,
        'True_Negative' : tn,
        'Accuracy' : round(mean(model_acc)*100, 4),
        'Precision' : round(mean(model_prec)*100, 4),
        'Recall' : round(mean(model_recall)*100, 4),
        'F1 Score' : round(mean(model_fscore), 4),
        'Specificity' : round(mean(model_spec)*100, 4),
        'MCC':round(mean(model_mcc), 4),
        'ROC_AUC_Score' :round(roc_auc_score(lst_actual_class,lst_predicted_prob_class,
multi_class='ovo', average='weighted'),3) ,
        'Balanced Accuracy': round(mean(model_bal_acc)*100, 4)}
    Results = Results.append(new_row, ignore_index=True)
```

4. Conclusions and Future work

Based on the analysis , keeping restaurant business in mind, I tried to figure out answers to some of the common queries when opening any new restaurant.

- I figured BTM, Koramangala, HSR are good places to start restaurant. Whitefield has the greatest number of unique restaurants and can be cheaper to get started. Koramangala, Indiranagar, BTM are most popular locations among foodies.
- Large number of votes can ensure better rating and 1K for 2 people is good to go price.
- Bangalorian love fast food.
- Providing online ordering can boast your chances.

The model results in the following order by considering the model accuracy, F1 score and RoC AUC score.

1) **Bagging Classifier**
2) **Random Forest**
3) **Extra Trees Classifier**

1) We recommend model - **Bagging Classifier** with Stratified and Random Sampling technique as a best fit for the given Zomato Bangalore dataset. We considered Bagging Classifier because it uses bootstrap aggregation which can reduce bias and variance in the data and can leads to good predictions with

	Model Name	True_Positive	False_Negative	False_Positive	True_Negative	Accuracy	Precision	Recall	F1 Score	Specificit
0	LogisticRegression(multi_class='multinomial', ...	923	286	860	8275	91.4	37.075	NaN	0.3782	90.37
1	DecisionTreeClassifier()	1705	60	78	8501	98.5	95.325	94.775	0.9502	98.
2	(DecisionTreeClassifier(max_features='auto', r...	1700	38	83	8523	98.625	92.725	97.825	0.95	98.77
3	(ExtraTreeClassifier(random_state=1130007044),...	1694	43	89	8518	98.475	92.4	96.95	0.9442	98.57
4	KNeighborsClassifier()	1637	175	146	8386	96.8	88.425	81.625	0.8415	95.87
5	GaussianNB()	1126	624	657	7937	90.75	63.825	NaN	0.4175	87.7
6	(DecisionTreeClassifier(random_state=171011636...	1723	36	60	8525	99.0	95.65	97.875	0.9672	98.9
7	([DecisionTreeRegressor(criterion='friedman_ms...	1237	317	546	8244	92.7	66.625	54.675	0.4438	91.
8	SVC(probability=True)	985	326	798	8235	91.525	37.775	NaN	0.3835	90.1
9	XGBClassifier(base_score=0.5, booster='gbtree'...	1364	311	419	8250	93.55	69.325	90.05	0.699	92.87
10	LGBMClassifier()	1514	147	269	8414	95.8	77.9	94.0	0.8158	96.1

claims dataset.

5. References

www.kaggle.com

www.github.com

https://medium.com/@vyshaghin/restaurant-success-prediction-analysis-and-model-building-42be18326397

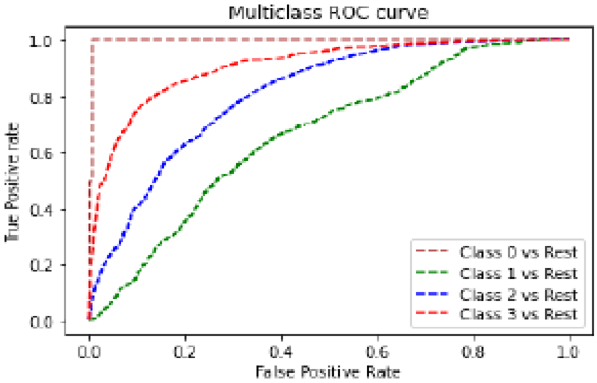
https://www.analyticsvidhya.com/blog/2020/08/types-of-categorical-data-encoding/

6.0. APPENDICES

- Data Modeling
- Data Mining
- Data Visualization
- Deep Learning
- Predictive Modeling
- Outliers
- Model Management and Training
- Clustering
- Big Data
- Supervised Learning
- Scaling

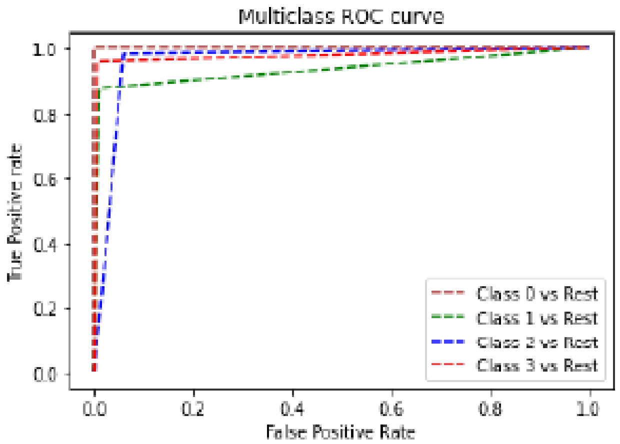
5.2. PYTHON CODE RESULTS

```
Model Name:  LogisticRegression(multi_class='multinomial', penalty='none',
random_state=42,
                   solver='newton-cg')
[[    0    0    0    0]
 [    0    0    1    1]
 [    2  627 7645  859]
 [    0    9  277  923]]
Print Class: 0
TP=0, FN=0, TN=10342, FP=2
Accuracy: 1.0
Precision: 0.0
Sensitivity: nan
F1-Score: 0.0
Specificity: 1.0
Balanced Accuracy: nan
MCC: nan

Print Class: 1
TP=0, FN=2, TN=9706, FP=636
Accuracy: 0.938
Precision: 0.0
Sensitivity: 0.0
F1-Score: 0.0
Specificity: 0.939
Balanced Accuracy: 0.47
MCC: -0.004

Print Class: 2
TP=7645, FN=1488, TN=933, FP=278
Accuracy: 0.829
Precision: 0.965
Sensitivity: 0.837
F1-Score: 0.896
Specificity: 0.77
Balanced Accuracy: 0.804
MCC: 0.461

Print Class: 3
TP=923, FN=286, TN=8275, FP=860
Accuracy: 0.889
Precision: 0.518
Sensitivity: 0.763
F1-Score: 0.617
Specificity: 0.906
Balanced Accuracy: 0.834
MCC: 0.569

Overall Performance Prediction:
Accuracy: 91.4%
Precision: 37.075%
Recall or Sensitivity: nan%
F1-Score: 0.3782
Specificity or True Nagative Rate: 90.375%
Balanced Accuracy: nan%
```

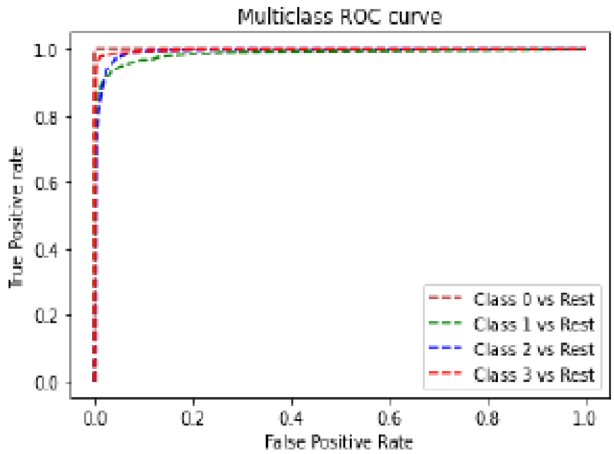



```
roc_auc_score: 0.818
--------------------------------------------------------------------
-------------------
Model Name:  DecisionTreeClassifier()
[[    2    0    0    0]
 [    0  557   98    6]
 [    0   73 7771   72]
 [    0    6   54 1705]]
Print Class: 0
TP=2, FN=0, TN=10342, FP=0
Accuracy: 1.0
Precision: 1.0
Sensitivity: 1.0
F1-Score: 1.0
Specificity: 1.0
Balanced Accuracy: 1.0
MCC: 1.0

Print Class: 1
TP=557, FN=104, TN=9604, FP=79
Accuracy: 0.982
Precision: 0.876
Sensitivity: 0.843
F1-Score: 0.859
Specificity: 0.992
Balanced Accuracy: 0.918
MCC: 0.85

Print Class: 2
TP=7771, FN=145, TN=2276, FP=152
Accuracy: 0.971
Precision: 0.981
Sensitivity: 0.982
F1-Score: 0.981
Specificity: 0.937
Balanced Accuracy: 0.96
MCC: 0.92

Print Class: 3
TP=1705, FN=60, TN=8501, FP=78
Accuracy: 0.987
```

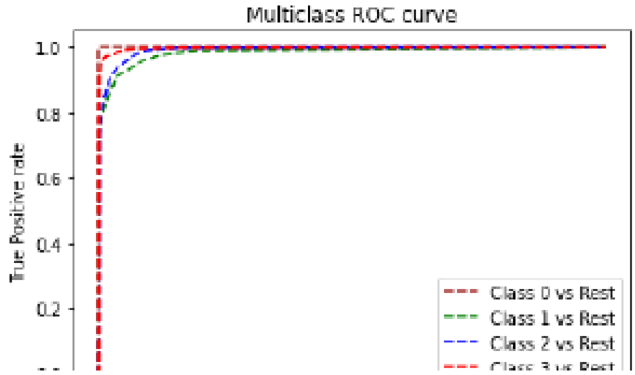
```
Precision: 0.956
Sensitivity: 0.966
F1-Score: 0.961
Specificity: 0.991
Balanced Accuracy: 0.978
MCC: 0.953

Overall Performance Prediction:
Accuracy: 98.5%
Precision: 95.325%
Recall or Sensitivity: 94.775%
F1-Score: 0.9502
Specificity or True Nagative Rate: 98.0%
Balanced Accuracy: 96.4%

MCC: 0.9308
```




```
roc_auc_score: 0.967
-----------------------------------------------------
-------------------
Model Name:  RandomForestClassifier()
[[   2    0    0    0]
 [   0  485   17    1]
 [   0  149 7870   82]
 [   0    2   36 1700]]
Print Class: 0
TP=2, FN=0, TN=10342, FP=0
Accuracy: 1.0
Precision: 1.0
Sensitivity: 1.0
F1-Score: 1.0
Specificity: 1.0
Balanced Accuracy: 1.0
MCC: 1.0

Print Class: 1
TP=485, FN=18, TN=9690, FP=151
Accuracy: 0.984
Precision: 0.763
Sensitivity: 0.964
F1-Score: 0.852
Specificity: 0.985
Balanced Accuracy: 0.974
MCC: 0.85

Print Class: 2
TP=7870, FN=231, TN=2190, FP=53
Accuracy: 0.973
Precision: 0.993
Sensitivity: 0.971
F1-Score: 0.982
Specificity: 0.976
Balanced Accuracy: 0.974
MCC: 0.923

Print Class: 3
TP=1700, FN=38, TN=8523, FP=83
Accuracy: 0.988
Precision: 0.953
Sensitivity: 0.978
F1-Score: 0.966
Specificity: 0.99
Balanced Accuracy: 0.984
MCC: 0.959

Overall Performance Prediction:
Accuracy: 98.625%
Precision: 92.725%
Recall or Sensitivity: 97.825%
F1-Score: 0.95
Specificity or True Nagative Rate: 98.775%
Balanced Accuracy: 98.3%

MCC: 0.933
```

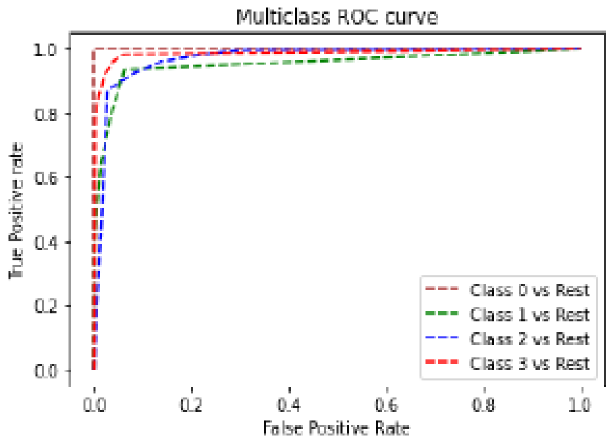


```
----------------------------------------------
--------------------
Model Name:  ExtraTreesClassifier()
[[   2    0    0    0]
 [   0  480   32    3]
 [   0  151 7853   86]
 [   0    5   38 1694]]
Print Class: 0
TP=2, FN=0, TN=10342, FP=0
Accuracy: 1.0
Precision: 1.0
Sensitivity: 1.0
F1-Score: 1.0
Specificity: 1.0
Balanced Accuracy: 1.0
MCC: 1.0

Print Class: 1
TP=480, FN=35, TN=9673, FP=156
Accuracy: 0.982
Precision: 0.755
Sensitivity: 0.932
F1-Score: 0.834
Specificity: 0.984
Balanced Accuracy: 0.958
MCC: 0.83

Print Class: 2
TP=7853, FN=237, TN=2184, FP=70
Accuracy: 0.97
Precision: 0.991
Sensitivity: 0.971
F1-Score: 0.981
Specificity: 0.969
Balanced Accuracy: 0.97
MCC: 0.916

Print Class: 3
TP=1694, FN=43, TN=8518, FP=89
Accuracy: 0.987
Precision: 0.95
Sensitivity: 0.975
F1-Score: 0.962
Specificity: 0.99
Balanced Accuracy: 0.982
MCC: 0.955

Overall Performance Prediction:
Accuracy: 98.475%
Precision: 92.4%
Recall or Sensitivity: 96.95%
F1-Score: 0.9442
Specificity or True Nagative Rate: 98.575%
Balanced Accuracy: 97.75%

MCC: 0.9252
```



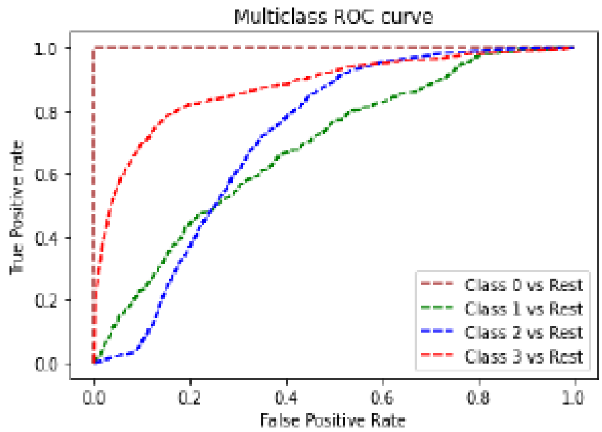

```
roc_auc_score: 0.993
----------------------------------------------------
---------------------
Model Name:  KNeighborsClassifier()
[[   2    0    1    0]
 [   0  418  132   17]
 [   0  209 7624  129]
 [   0    9  166 1637]]
Print Class: 0
TP=2, FN=1, TN=10341, FP=0
Accuracy: 1.0
Precision: 1.0
Sensitivity: 0.667
F1-Score: 0.8
Specificity: 1.0
Balanced Accuracy: 0.834
MCC: 0.816

Print Class: 1
TP=418, FN=149, TN=9559, FP=218
Accuracy: 0.965
Precision: 0.657
Sensitivity: 0.737
F1-Score: 0.695
Specificity: 0.978
Balanced Accuracy: 0.857
MCC: 0.677

Print Class: 2
TP=7624, FN=338, TN=2083, FP=299
Accuracy: 0.938
Precision: 0.962
Sensitivity: 0.958
F1-Score: 0.96
Specificity: 0.874
Balanced Accuracy: 0.916
MCC: 0.827

Print Class: 3
TP=1637, FN=175, TN=8386, FP=146
Accuracy: 0.969
Precision: 0.918
Sensitivity: 0.903
F1-Score: 0.911
Specificity: 0.983
Balanced Accuracy: 0.943
MCC: 0.892
```

```
Overall Performance Prediction:
Accuracy: 96.8%
Precision: 88.425%
Recall or Sensitivity: 81.625%
F1-Score: 0.8415
Specificity or True Nagative Rate: 95.875%
Balanced Accuracy: 88.75%

MCC: 0.803
```




```
roc_auc_score: 0.977
--------------------------------------------------------------------------------
Model Name:  GaussianNB()
[[    2    4   14    5]
 [    0    0    0    0]
 [    0  620 7297  652]
 [    0   12  612 1126]]
Print Class: 0
TP=2, FN=23, TN=10319, FP=0
Accuracy: 0.998
Precision: 1.0
Sensitivity: 0.08
F1-Score: 0.148
Specificity: 1.0
Balanced Accuracy: 0.54
MCC: 0.283

Print Class: 1
TP=0, FN=0, TN=9708, FP=636
Accuracy: 0.939
Precision: 0.0
Sensitivity: nan
F1-Score: 0.0
Specificity: 0.939
Balanced Accuracy: nan
MCC: nan
```

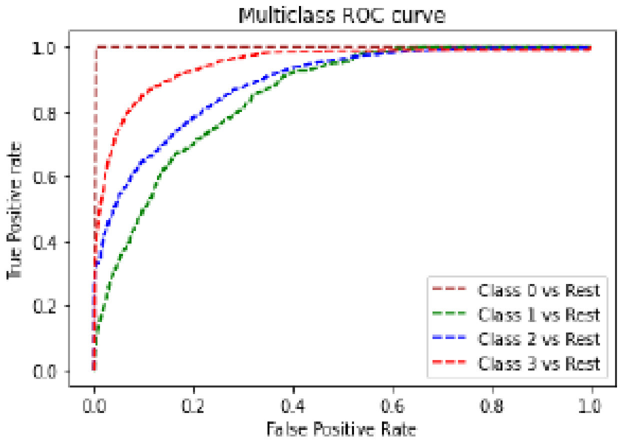
```
TP=7297, FN=1272, TN=1149, FP=626
Accuracy: 0.817
Precision: 0.921
Sensitivity: 0.852
F1-Score: 0.885
Specificity: 0.647
Balanced Accuracy: 0.75
MCC: 0.444

Print Class: 3
TP=1126, FN=624, TN=7937, FP=657
Accuracy: 0.876
Precision: 0.632
Sensitivity: 0.643
F1-Score: 0.637
Specificity: 0.924
Balanced Accuracy: 0.784
MCC: 0.563

Overall Performance Prediction:
Accuracy: 90.75%
Precision: 63.825%
Recall or Sensitivity: nan%
F1-Score: 0.4175
Specificity or True Nagative Rate: 87.75%
Balanced Accuracy: nan%

MCC: nan
```




```
roc_auc_score: 0.819
--------------------------------------------------------------------------------
Model Name:  BaggingClassifier(n_estimators=100)
[[    2    0    0    0]
 [    0  552   26    1]
```

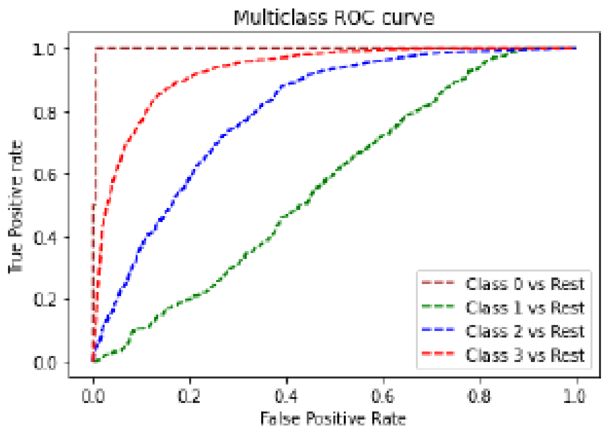
```
 [    0   83 7862   59]
 [    0    1   35 1723]]
Print Class: 0
TP=2, FN=0, TN=10342, FP=0
Accuracy: 1.0
Precision: 1.0
Sensitivity: 1.0
F1-Score: 1.0
Specificity: 1.0
Balanced Accuracy: 1.0
MCC: 1.0

Print Class: 1
TP=552, FN=27, TN=9681, FP=84
Accuracy: 0.989
Precision: 0.868
Sensitivity: 0.953
F1-Score: 0.909
Specificity: 0.991
Balanced Accuracy: 0.972
MCC: 0.904

Print Class: 2
TP=7862, FN=142, TN=2279, FP=61
Accuracy: 0.98
Precision: 0.992
Sensitivity: 0.982
F1-Score: 0.987
Specificity: 0.974
Balanced Accuracy: 0.978
MCC: 0.945

Print Class: 3
TP=1723, FN=36, TN=8525, FP=60
Accuracy: 0.991
Precision: 0.966
Sensitivity: 0.98
F1-Score: 0.973
Specificity: 0.993
Balanced Accuracy: 0.986
MCC: 0.967

Overall Performance Prediction:
Accuracy: 99.0%
Precision: 95.65%
Recall or Sensitivity: 97.875%
F1-Score: 0.9672
Specificity or True Nagative Rate: 98.
Balanced Accuracy: 98.4%

MCC: 0.954
```

```
F1-Score: 0.4438
Specificity or True Nagative Rate: 91.7%
Balanced Accuracy: 73.2%

MCC: 0.3885
```




```
roc_auc_score: 0.916
-----------------------------------------------------------
------------------
Model Name:  SVC(probability=True)
[[    0    0    0    0]
 [    0    0    0    0]
 [    1  632 7602  798]
 [    1    4  321  985]]
Print Class: 0
TP=0, FN=0, TN=10342, FP=2
Accuracy: 1.0
Precision: 0.0
Sensitivity: nan
F1-Score: 0.0
Specificity: 1.0
Balanced Accuracy: nan
MCC: nan

Print Class: 1
TP=0, FN=0, TN=9708, FP=636
Accuracy: 0.939
Precision: 0.0
Sensitivity: nan
F1-Score: 0.0
Specificity: 0.939
Balanced Accuracy: nan
MCC: nan

Print Class: 2
TP=7602, FN=1431, TN=990, FP=321
Accuracy: 0.831
```

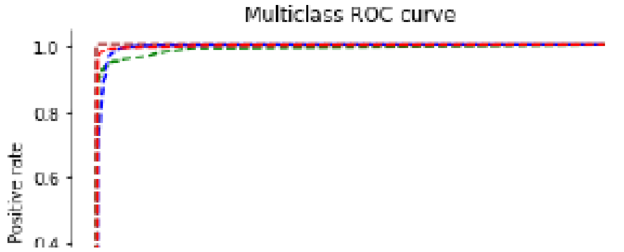
```
Precision: 0.959
Sensitivity: 0.842
F1-Score: 0.897
Specificity: 0.755
Balanced Accuracy: 0.798
MCC: 0.469

Print Class: 3
TP=985, FN=326, TN=8235, FP=798
Accuracy: 0.891
Precision: 0.552
Sensitivity: 0.751
F1-Score: 0.637
Specificity: 0.912
Balanced Accuracy: 0.832
MCC: 0.584

Overall Performance Prediction:
Accuracy: 91.525%
Precision: 37.775%
Recall or Sensitivity: nan%
F1-Score: 0.3835
Specificity or True Nagative Rate: 90.15%
Balanced Accuracy: nan%

MCC: nan
```



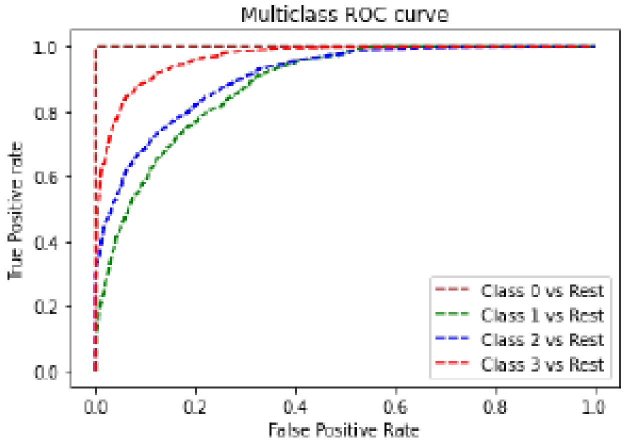

```
roc_auc_score: 0.81
```

```
[[    2     0     0     0]
 [    0    29     2     1]
 [    0   598  7619   418]
 [    0     9   302  1364]]
Print Class: 0
TP=2, FN=0, TN=10342, FP=0
Accuracy: 1.0
Precision: 1.0
Sensitivity: 1.0
F1-Score: 1.0
Specificity: 1.0
Balanced Accuracy: 1.0
MCC: 1.0

Print Class: 1
TP=29, FN=3, TN=9705, FP=607
Accuracy: 0.941
Precision: 0.046
Sensitivity: 0.906
F1-Score: 0.087
Specificity: 0.941
Balanced Accuracy: 0.924
MCC: 0.196

Print Class: 2
TP=7619, FN=1016, TN=1405, FP=304
Accuracy: 0.872
Precision: 0.962
Sensitivity: 0.882
F1-Score: 0.92
Specificity: 0.822
Balanced Accuracy: 0.852
MCC: 0.618

Print Class: 3
TP=1364, FN=311, TN=8250, FP=419
Accuracy: 0.929
Precision: 0.765
Sensitivity: 0.814
F1-Score: 0.789
Specificity: 0.952
Balanced Accuracy: 0.883
MCC: 0.747

Overall Performance Prediction:
Accuracy: 93.55%
Precision: 69.325%
Recall or Sensitivity: 90.05%
F1-Score: 0.699
Specificity or True Nagative Rate: 92.875%
Balanced Accuracy: 91.475%

MCC: 0.6402
```

```
---------------------------------------------------------------------------
-------------------
Model Name:  XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric='mlogloss', gamma=0, gpu_id=-1,
              grow_policy='depthwise', importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_bin=256, max_cat_to_onehot=4, max_delta_step=0, max_depth=
3.
```




```
roc_auc_score: 0.994
------------------------------------------------
---------------------
Model Name:  GradientBoostingClassifier()
[[   2    0   22   15]
 [   0    8    8    1]
 [   0  617 7587  530]
 [   0   11  306 1237]]
Print Class: 0
TP=2, FN=37, TN=10305, FP=0
Accuracy: 0.996
Precision: 1.0
Sensitivity: 0.051
F1-Score: 0.098
Specificity: 1.0
Balanced Accuracy: 0.526
MCC: 0.226

Print Class: 1
TP=8, FN=9, TN=9699, FP=628
Accuracy: 0.938
Precision: 0.013
Sensitivity: 0.471
F1-Score: 0.025
Specificity: 0.939
Balanced Accuracy: 0.705
MCC: 0.069

Print Class: 2
TP=7587, FN=1147, TN=1274, FP=336
Accuracy: 0.857
Precision: 0.958
Sensitivity: 0.869
F1-Score: 0.911
Specificity: 0.791
Balanced Accuracy: 0.83
MCC: 0.565

Print Class: 3
TP=1237, FN=317, TN=8244, FP=546
Accuracy: 0.917
Precision: 0.694
Sensitivity: 0.796
F1-Score: 0.741
Specificity: 0.938
Balanced Accuracy: 0.867
MCC: 0.694

Overall Performance Prediction:
Accuracy: 92.7%
Precision: 66.625%
Recall or Sensitivity: 54.675%
```

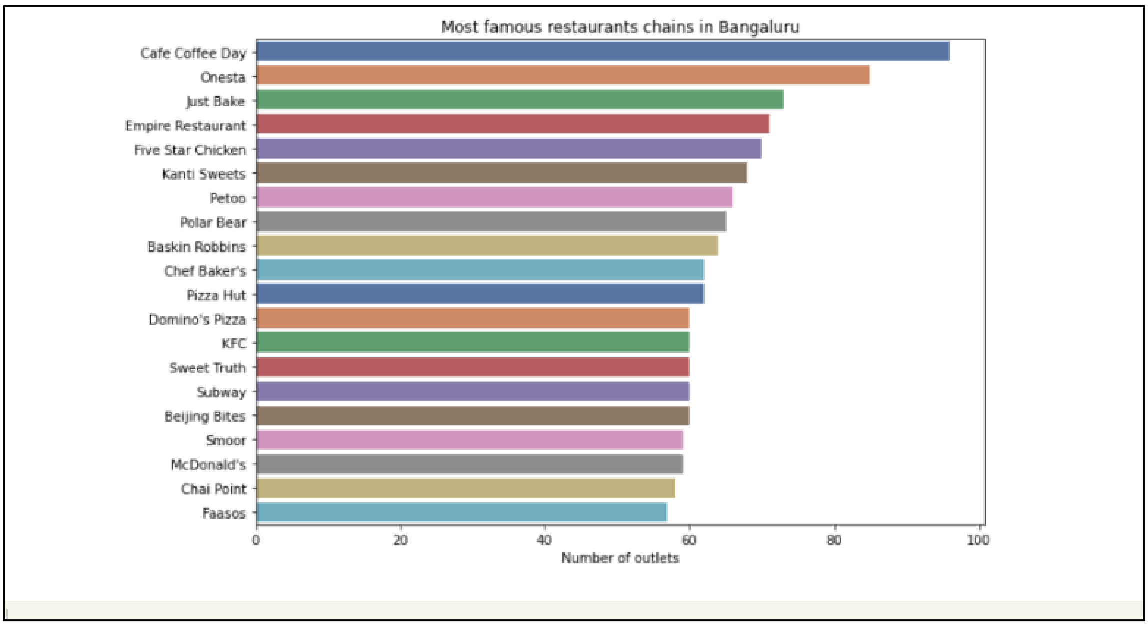



```
roc_auc_score: 0.933
-----------------------------------------------------
-------------------
Model Name:  LGBMClassifier()
[[   2    0    0    0]
 [   0  181   11    2]
 [   0  442 7778  267]
 [   0   13  134 1514]]
Print Class: 0
TP=2, FN=0, TN=10342, FP=0
Accuracy: 1.0
Precision: 1.0
Sensitivity: 1.0
F1-Score: 1.0
Specificity: 1.0
Balanced Accuracy: 1.0
MCC: 1.0

Print Class: 1
TP=181, FN=13, TN=9695, FP=455
Accuracy: 0.955
Precision: 0.285
Sensitivity: 0.933
F1-Score: 0.436
Specificity: 0.955
Balanced Accuracy: 0.944
MCC: 0.502

Print Class: 2
TP=7778, FN=709, TN=1712, FP=145
Accuracy: 0.917
Precision: 0.982
Sensitivity: 0.916
F1-Score: 0.948
Specificity: 0.922
Balanced Accuracy: 0.919
MCC: 0.76

Print Class: 3
```

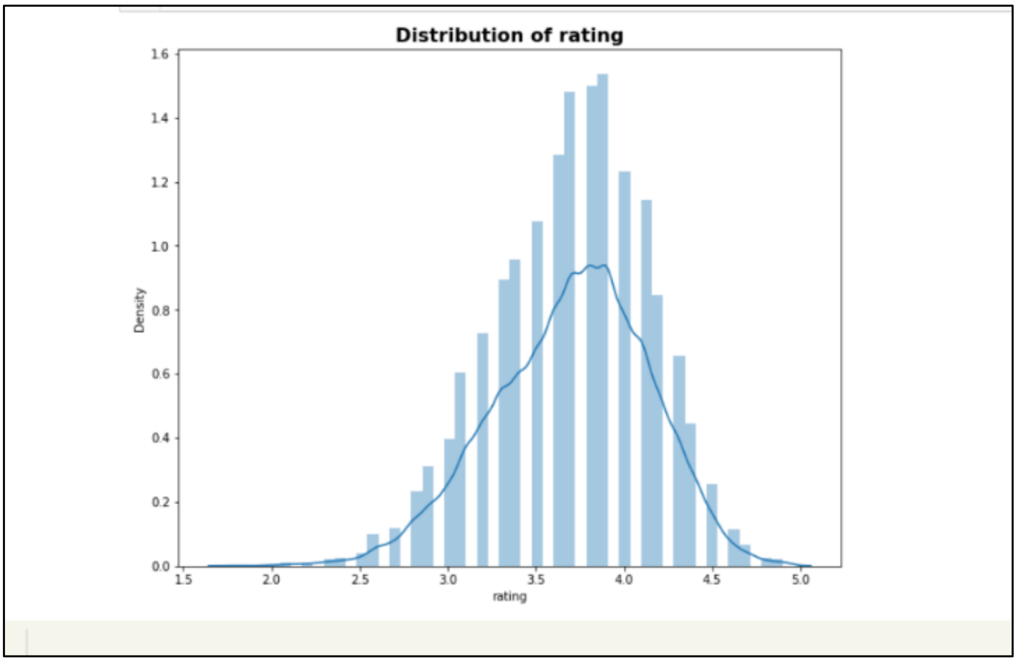
Code Results of different Models

5.3. List of Charts

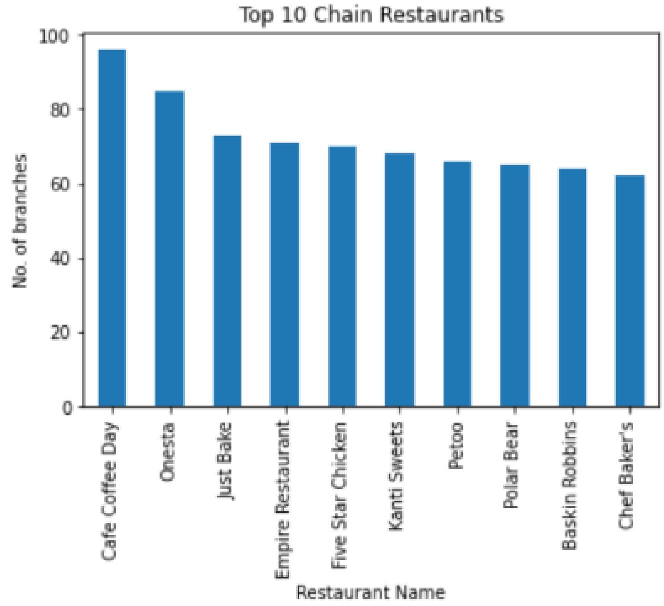
5.3.1. Chart 01: Top chain Restaurants in Bengaluru



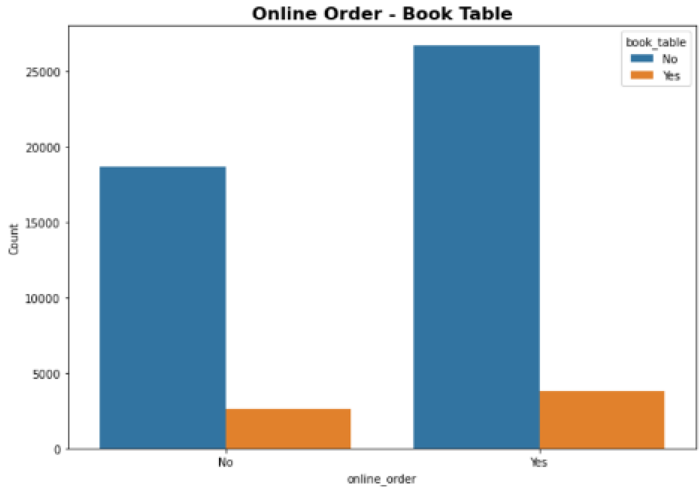
5.3.2. Chart 02: Distribution of Rating



5.3.3. Chart 03: Most Popular Restaurant Chains



5.3.4. Chart 04: Online Order vs Table Booking



Observation: We observe that maximum restaurants provide online ordering but not table booking facility. The number of restaurants providing table booking facility but not online order is the least. More than 15000 restaurants don't provide online ordering as well as table booking facility.