Team:

Subha Bose

Project Outline:

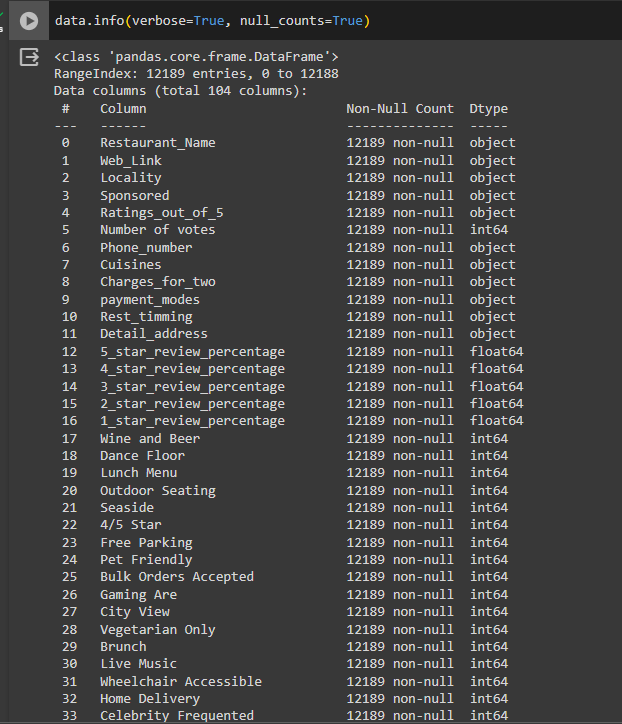
Restaurant Rating Determinants and Recommendation System.

Description:

This project aims to create a recommendation system for restaurants in Pune, India, while also examining the factors that impact restaurant ratings. Leveraging Python programming language, we adhere to Object-Oriented Programming (OOP) principles to build a modular and scalable codebase. Utilizing databases, particularly relational databases managed through SQL, we efficiently store and manage restaurant data. The recommendation system employs collaborative filtering algorithms, implemented using Python's machine learning libraries like sci-kit-learn. Simultaneously, we conduct an in-depth analysis of various factors influencing restaurant ratings using statistical techniques and data visualization tools available in Python. This approach ensures the development of a comprehensive solution that offers personalized recommendations while shedding light on the determinants of restaurant success in Pune.

Data Features:

* Covers ~12,000 restaurants in Pune.
* 104 diverse features
* Dataset source: Zomato Restaurants from Kaggle
* dtypes: float64(5), int64(87), object(12)
* memory usage: 9.7+ MB



Challenges:

While coming up with a recommendation system, it is important to go through a data of a considerable size to make a successful decision. However while dealing with data of such a size we are bound to come across a few challenges. Few of the challenges are:

1. Data Acquisition and Collection: Collecting massive amounts of data from various sources can be challenging, requiring careful planning and implementation of data acquisition pipelines to ensure data integrity and reliability.
2. Data Quality: Maintaining data quality becomes increasingly difficult as datasets grow in size. Data may be incomplete, inconsistent, or contain errors, necessitating thorough data cleaning and validation processes.
3. Data Exploration and Visualization: Exploring and visualizing large datasets can be challenging due to the sheer volume of data. Techniques such as data summarization, sampling, and interactive visualizations help extract meaningful insights from large datasets.
4. Model Complexity: Building models on large datasets requires careful consideration of model complexity and scalability. Complex models may be computationally expensive and prone to overfitting, necessitating techniques such as regularization and model simplification.

Current Progress:

The current status of the program involves the below steps with its code snippets being added to it

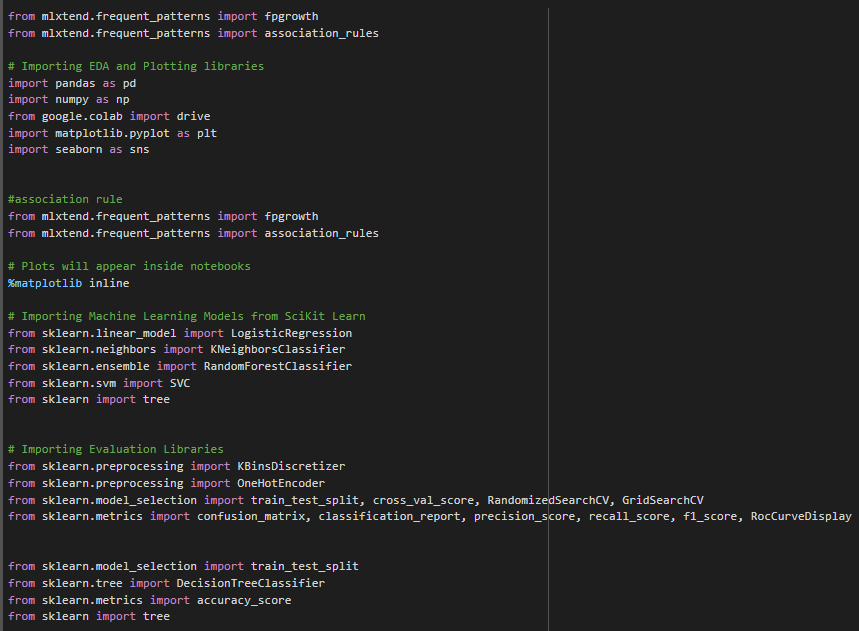
Step 1:

Mounting to drive to extract data: In order to extract data from a particular source on the machine.



Step 2:

Using Libraries: As we went along to design the code, we have incorporated various libraries and packages in order to save us on lines of coding and frankly to make use of Python’s amazing libraries.



Step 3:

Loading Excel File: Loading our data source which can be accessed by our code.



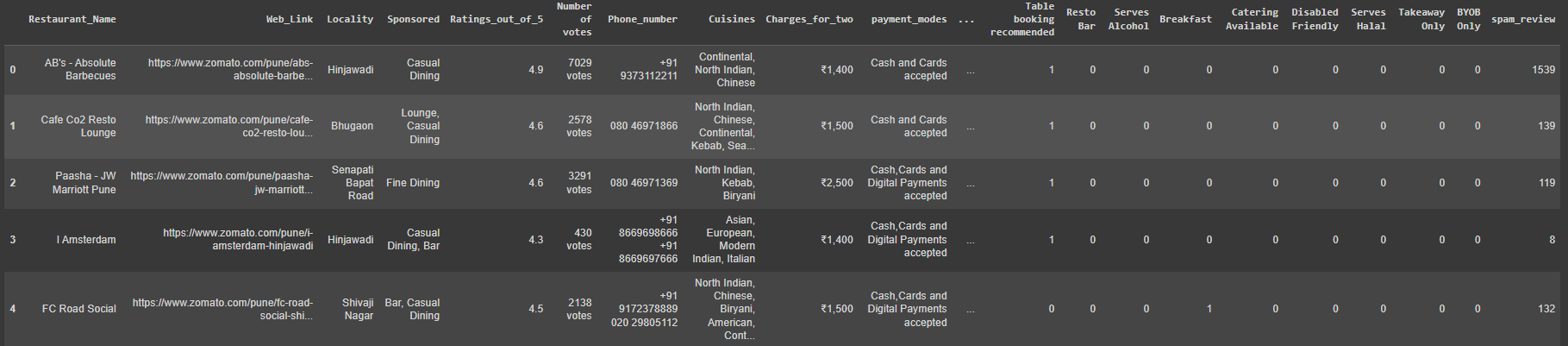
Step 4:

Viewing the dataset and all the attributes: In order to verify and to have a successful look into our loaded data

Code:



Result:



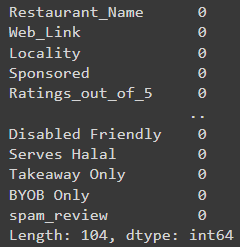
Step 5:

EDA: Exploratory Data Analysis, We begin our first step to clean the data and the foremost is to find the null values.

Code:



Result:



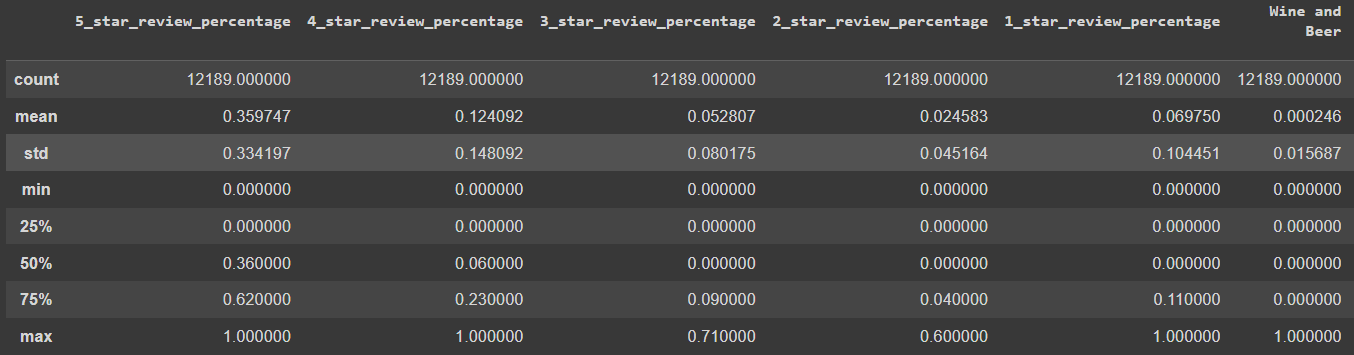
Step 6:

Statistical Analysis: In order to have a statistical overview of our data and to get basic idea using numbers and distribution.

Code:



Result:





Step 7:

Correlation: We have made use of correlation matrix to measure the strength and direction of the relationship between two columns of data representing different factors.

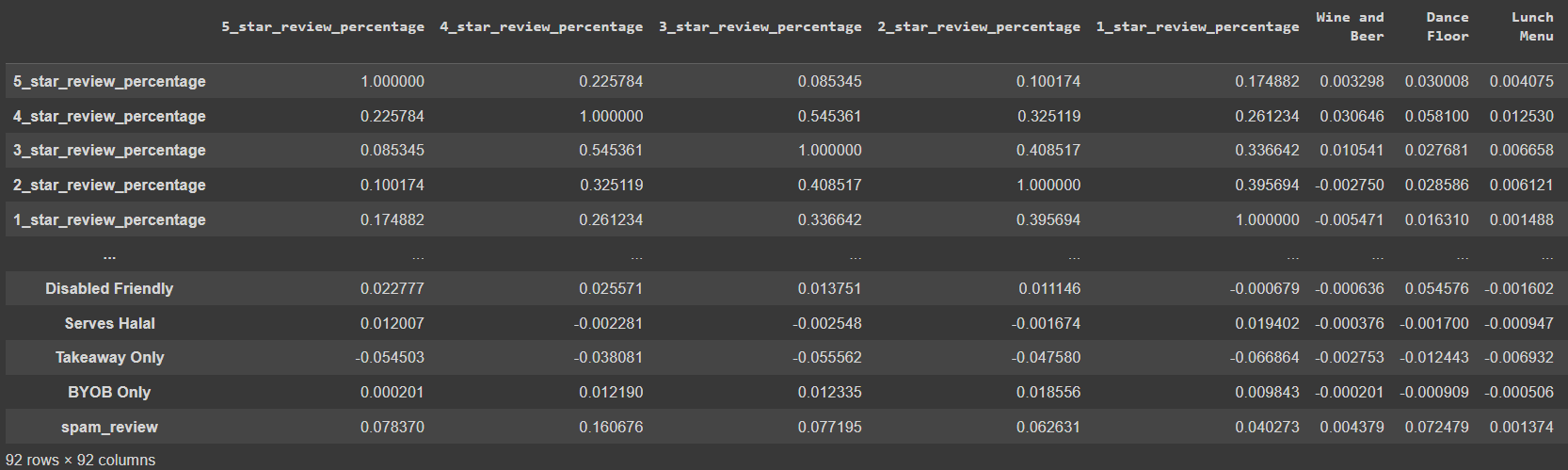
Code:



Result**:**

Correlation:

4 star restaurants have a high correlation with 3 star ratings.



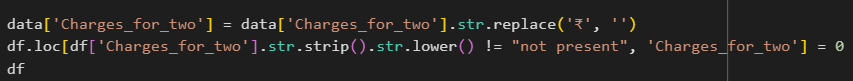
Step 8:

**One Hot Coding**:

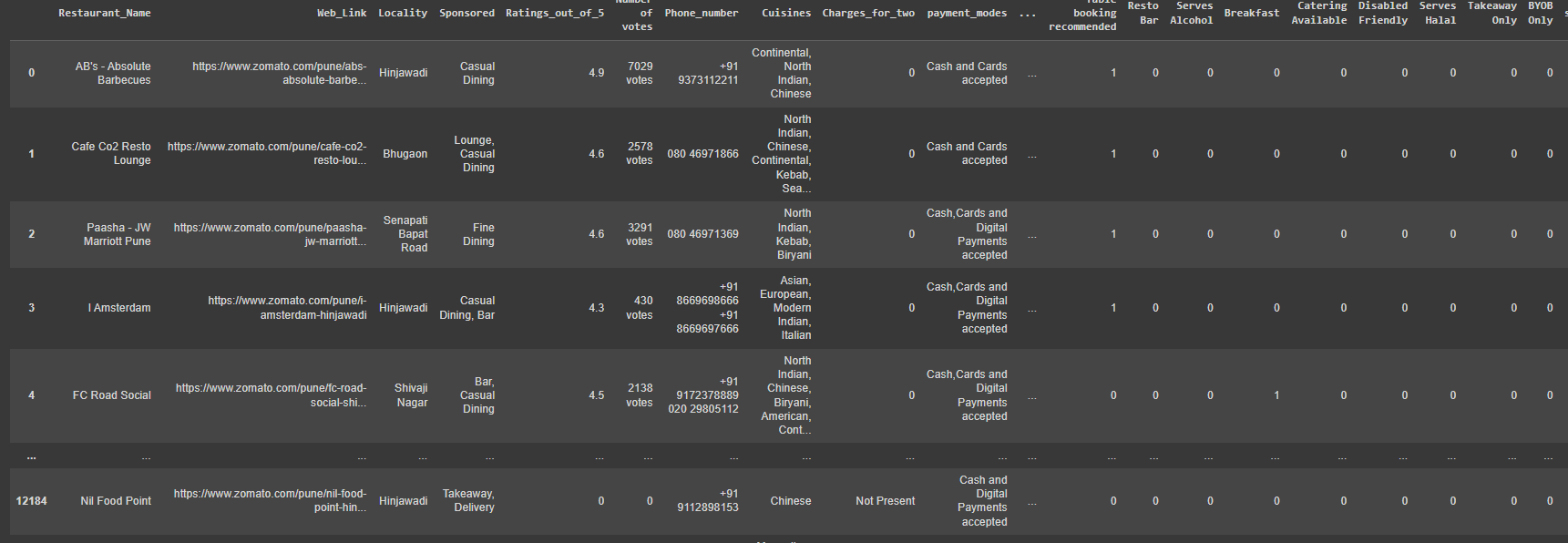
In order to make Machine Learning Models, need to make one-hot coded for attributes.

drop restaurant\_name, web\_link columns as it is not required (Unrequired Categorical Column) in Machine Learning models

Code:



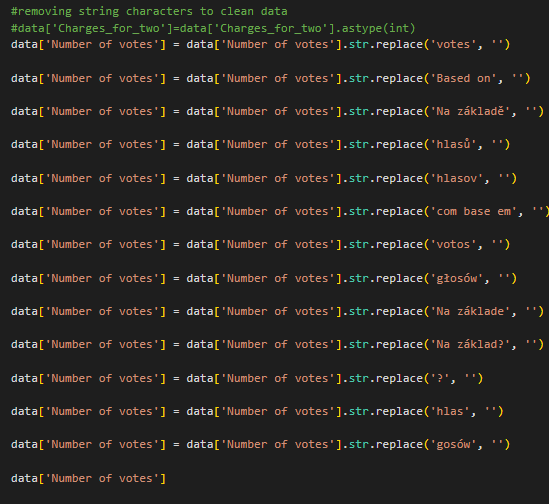
Result:



Step 9:

**Cleaning Data:** To be able to perform analysis of any kind, it is important to have data in a single format in order to perform correct analysis on it

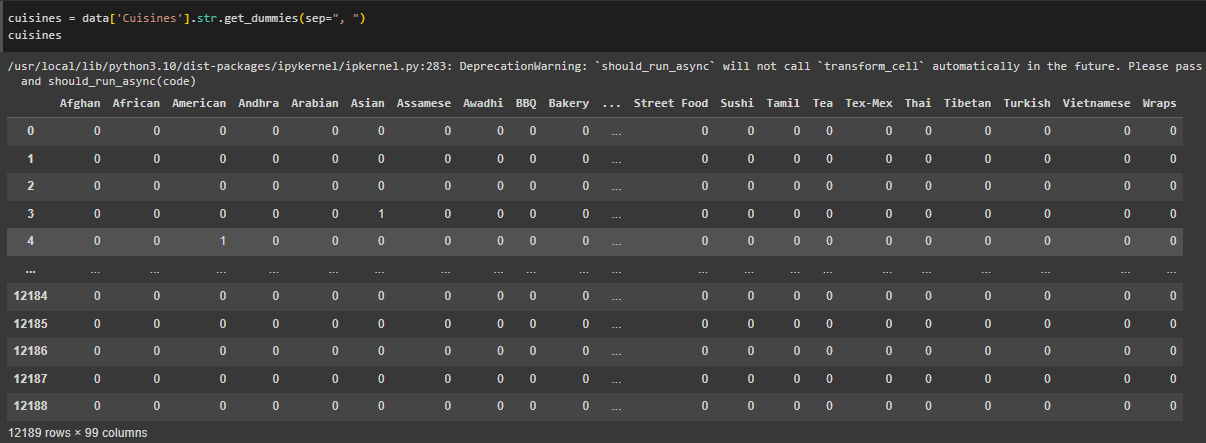
Code:



Step 10: **Data Preprocessing**

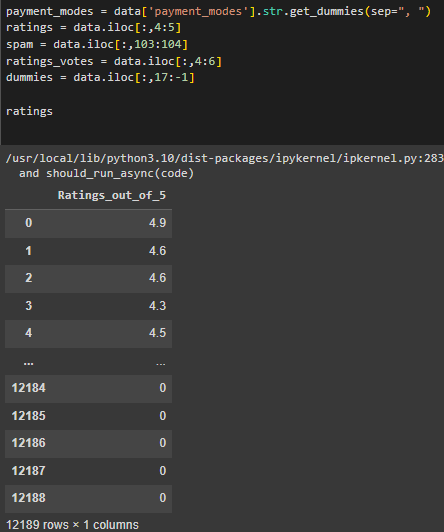
In order to make a decision tree we need to classify data in order to form decision tree paths

Here we are using the classifier CUISINES in order to better segregate the types of restaurants and keeping this information in order to use this feature for future predictions



Step 11: Data Preprocessing

The most important feature used to classify the restaurants is the RATING. So in order to classify how the restaurants are ordered we need to categorize them out of 5 which the best the customers can give to the restaurant.



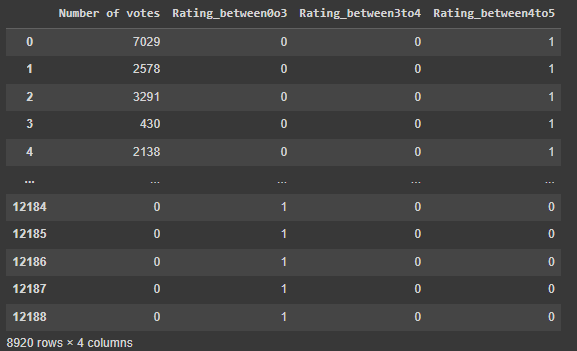
Step 12:**Data Preprocessing: One hot Encoding**

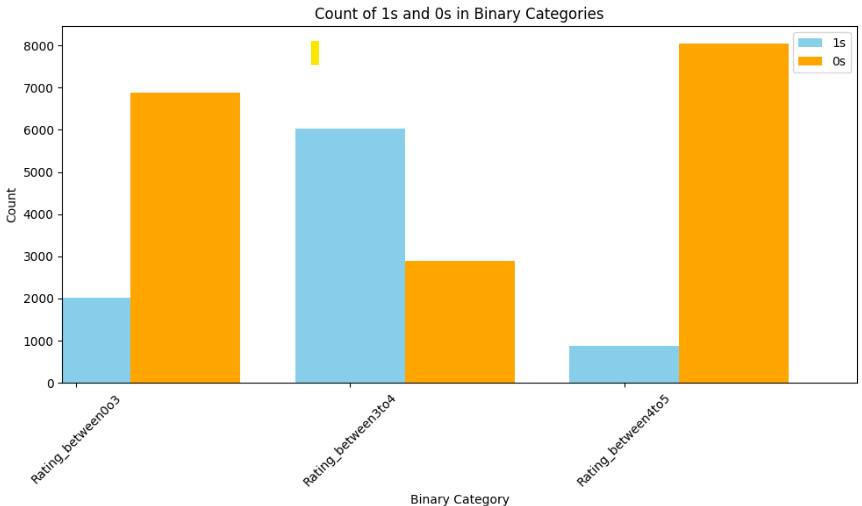
We plot the graph in order to show the

1: {**Yes**} The restaurant is placed is placed inside the Rating Bracket

0: {**No**} The restaurant is NOT placed is placed inside the Rating Bracket

As we plot the visualization it gives us the following result and table





Step 13: **Data Wrangling** for analyzing data

Till now we have classified all the restaurants into different categories and based on the results we perform more machine learning algorithms to yield better results

Observations:

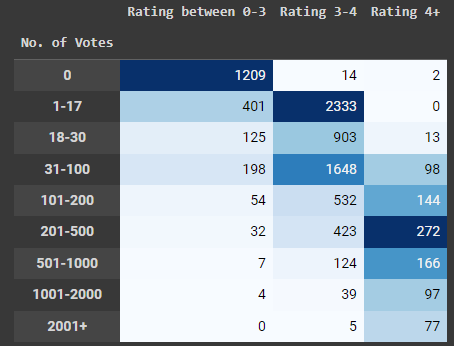
If number feedback of **votes** = **0** then rating of restaurant is **less than 3** for the entire dataset

As the votes increase: **Votes = {1-17**} **; Rating 3-4** => majority of **ratings** are **between 3-4**

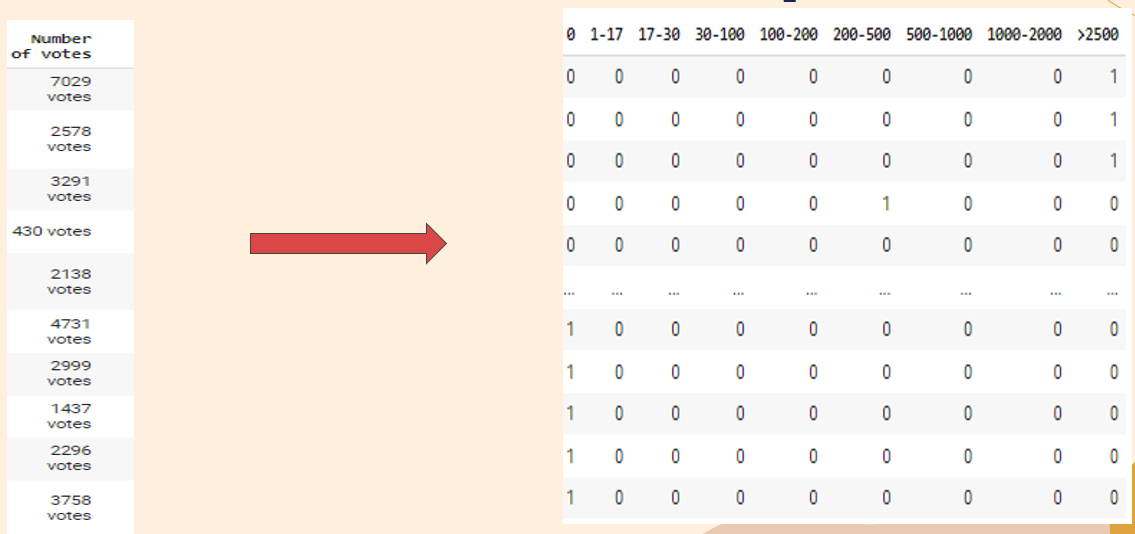
As thevotes increase: **Votes = {2001+) =>** majority votes **above 4**

Inference:

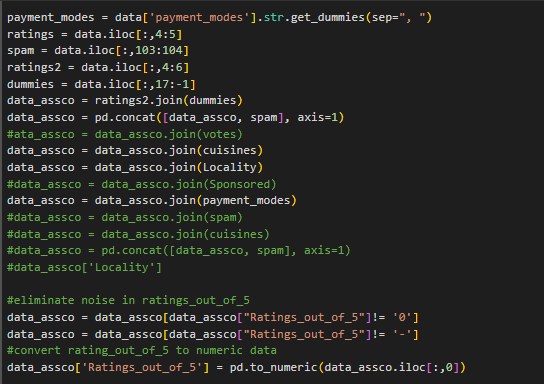
When the feedback vote count is low, then ratings are below 3 indicating less popular restaurants. As the count of feedback increases the votes tend to go Over 4, helping us in prediction by making a good decision tree on this data along with other features.

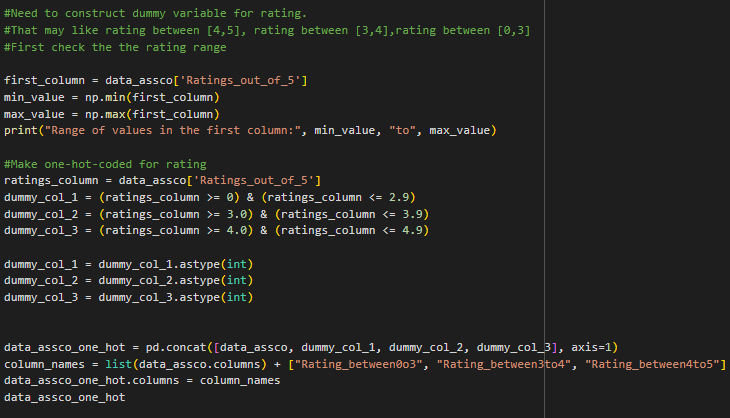


Data Transformation: Based above formulation (increased accuracy from 60% to 90%) on an average.

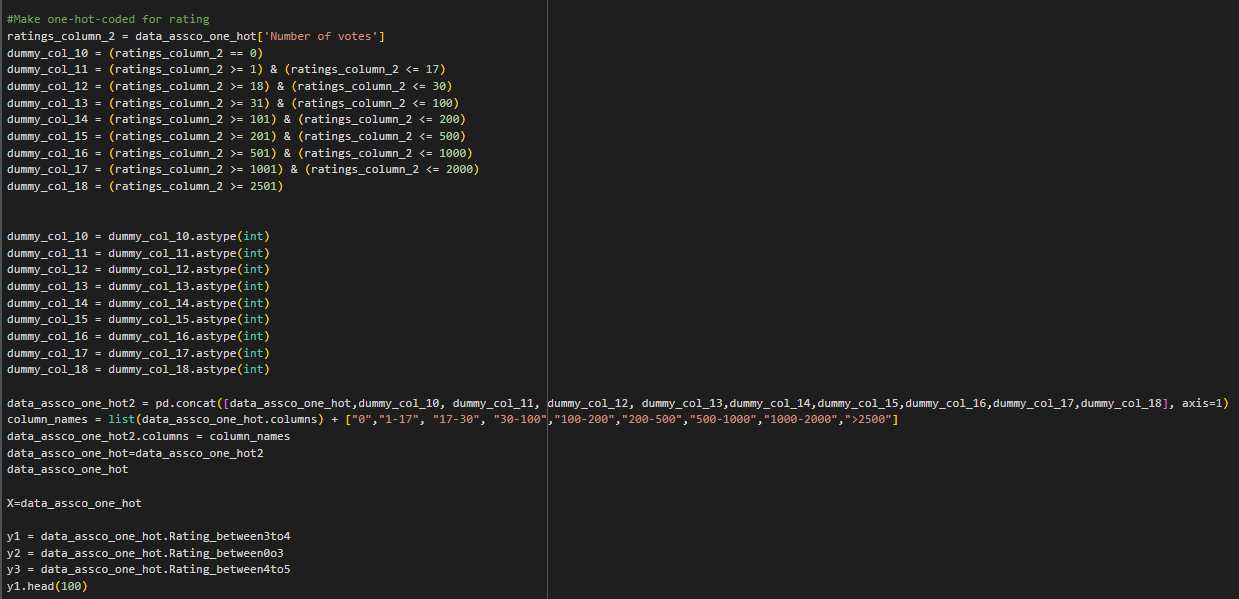


**Step 14**: **Using Pandas for data transformation** : This step performs various data preprocessing steps and feature engineering techniques on a dataset related to restaurant ratings and reviews

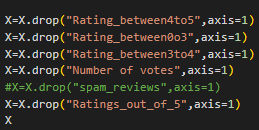


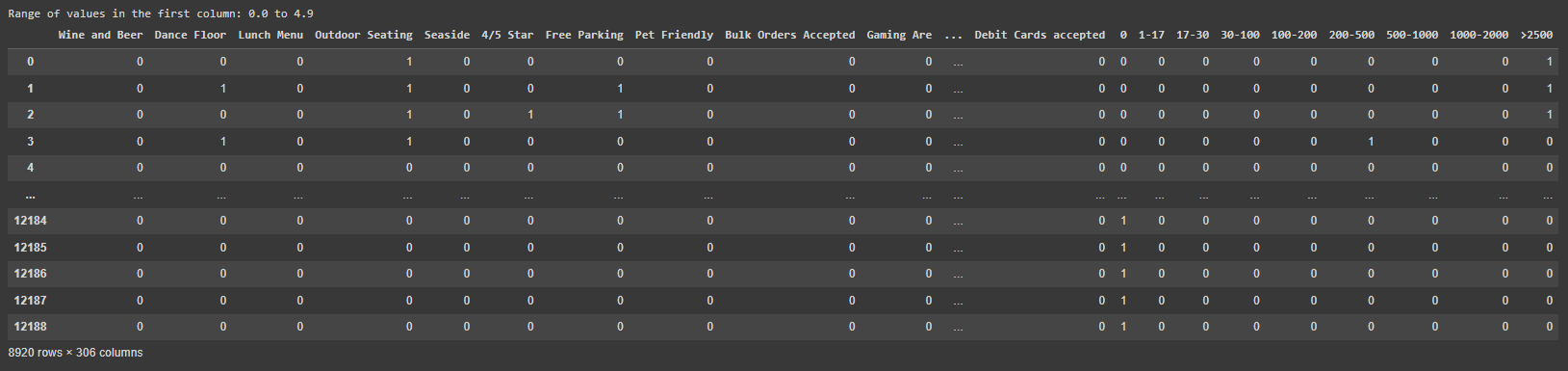


Y1, Y2, Y3: Creating 3 dependent variables for prediction (Rating above 4, between 3-4 and below 3)



Dropping categorical and not required attributes.





About above data frame:

X=(”**No of votes**”,“OutdoorSeating”, “VegetarianOnly”, “LiveMusic”, “HomeDelivery”, “Wifi”, “Buffet”, “TableReservationNotRequired”, “LiveSportsScreening”, “DessertsandBakes”, “MallParking”, “ValetParkingAvailable”, “KidFriendly”, “StandingTables”, “DeliveryOnly”, “LGBTQIAFriendly”, “Tablebookingrecommend”, “Brunch”) have significant positive relation to the “Restaurant ratings”

Y(labels to predict) : **Rating above 4**, **Rating between 3-4**, **Rating less than 3**

Data Split:X\_train, X\_test, y1\_train, y1\_test = train\_test\_split(X, y1, test\_size=0.7, random\_state=101)

X\_train, Y\_train: 2676 rows (train: 30%)

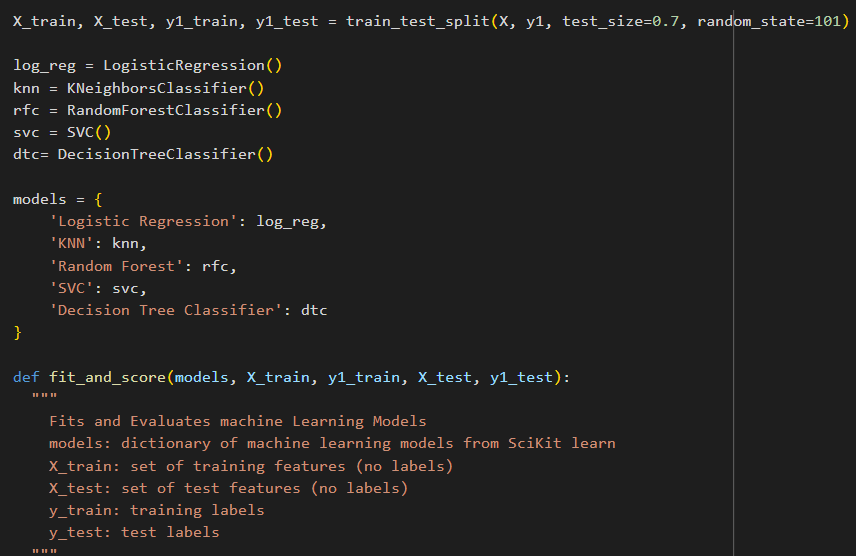
X\_test, Y\_test: 6244 rows (test: 70%)



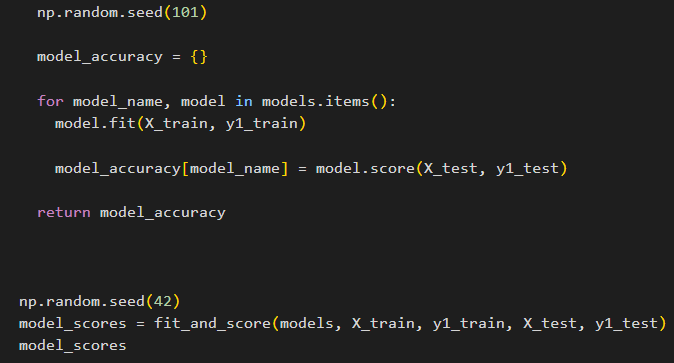
**Step 15: Training the data set:**

Here we train the model using the data we have pre-processed:

Creating a dictionary to store all 5 algorithms used for classification and storing accuracy values using **FOR** loop in the dictionary.



Seed= 101 for fixed random state



**Step 16: Model Comparison**

Here we are comparing the data on basis of accuracy with other Models

**Rating between 3-4**

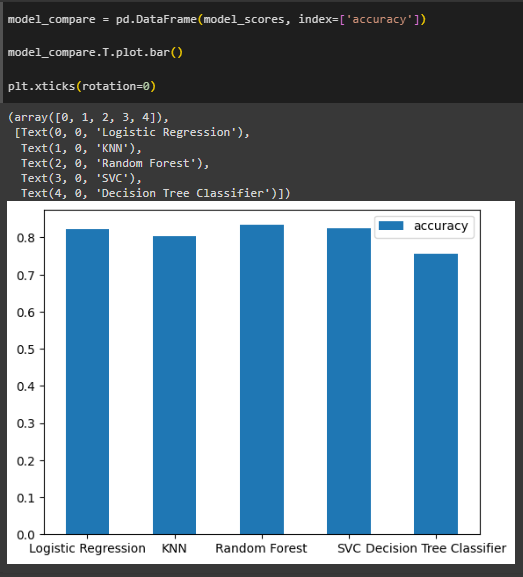
● LogisticRegression: 0.82

● Knn: 0.81

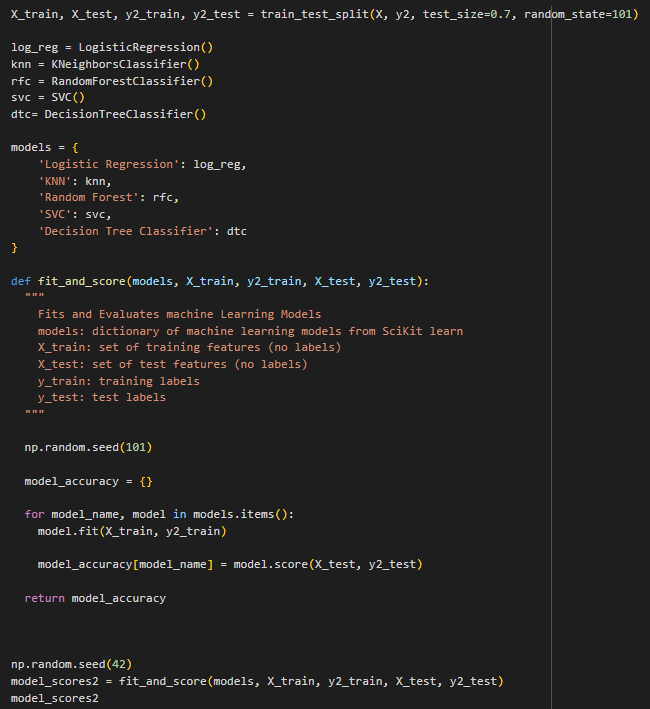
● RandomForestClassifier: 0.836

● SVC: 0.82

● Decision Tree: 0.758



**Step 17**: Comparing Training and Test Scores while increasing the number of neighbors in K nearest neighbors



**Step 18:** Comparing the scores for

**Rating above 4**

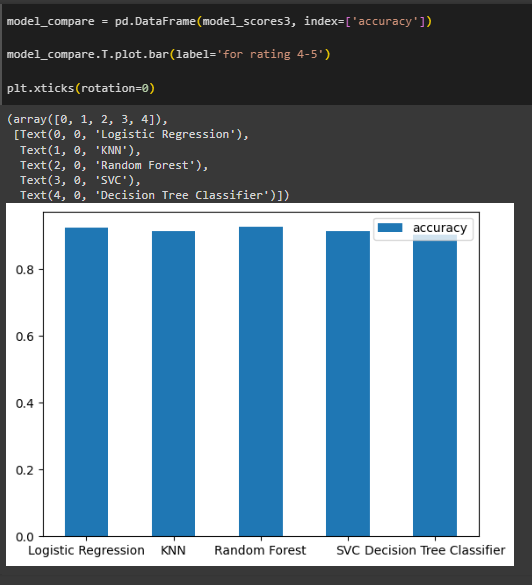
● LogisticRegression: 0.925

● Knn: 0.912

● RandomForestClassifier: 0.926

● SVC: 0.913

● Decision Tree: 0.903



**Step 19:** Performing a k-nearest neighbors (KNN) classification algorithm and then plotting the training and testing accuracy scores for different values of the number of neighbors (k).

train\_scores = []

test\_scores = []

neighbors = range(1, 21)

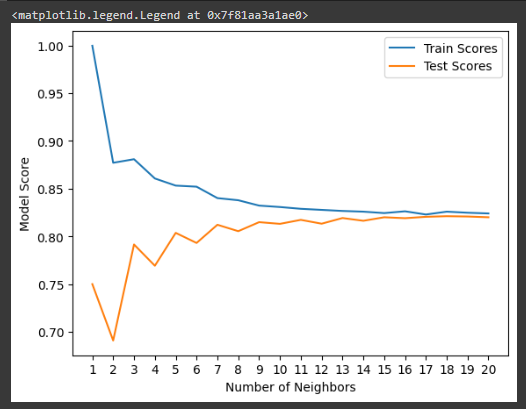
for k in neighbors:

knn = KNeighborsClassifier(n\_neighbors=k)

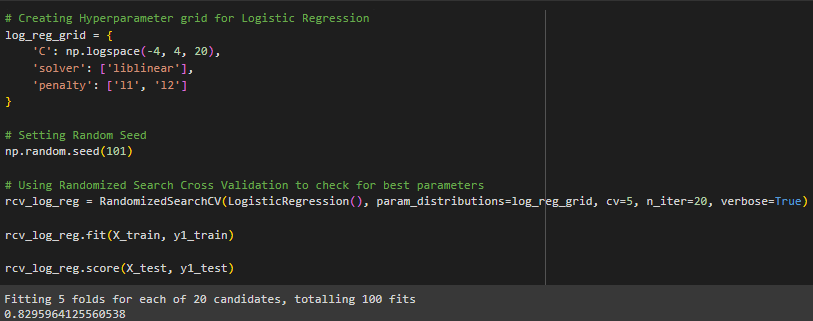
knn.fit(X\_train, y1\_train)

train\_scores.append(knn.score(X\_train, y1\_train))

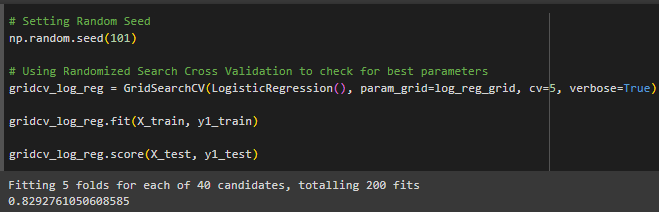
test\_scores.append(knn.score(X\_test, y1\_test))



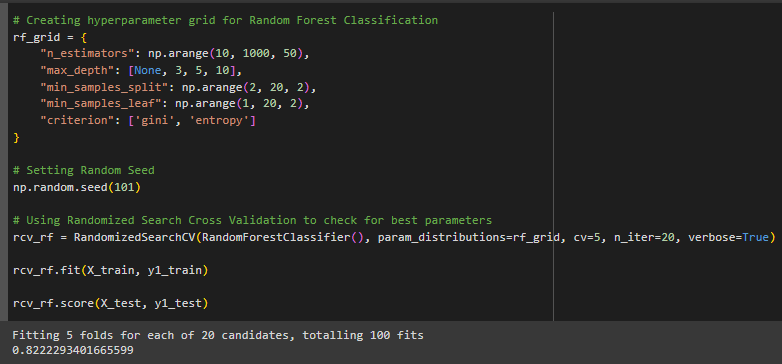
**Step 20**: Hyperparameter tuning for a Logistic Regression model using Randomized Search cross-validation (RandomizedSearchCV)



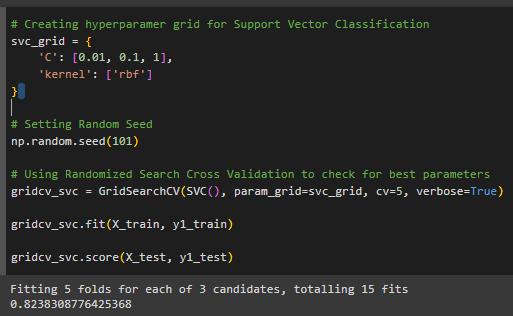
**Step 21**: performs hyperparameter tuning for a Logistic Regression model using Grid Search Cross Validation (GridSearchCV).



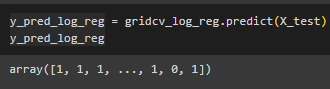
**Step 22**: Creating hyperparameterr grid for Random Forest Classification



**Step 23**: Creating hyperparamer grid for Support Vector Classification



**Step 24**: predictions using a Logistic Regression model that has been trained and tuned with hyperparameters using GridSearchCV (or RandomizedSearchCV

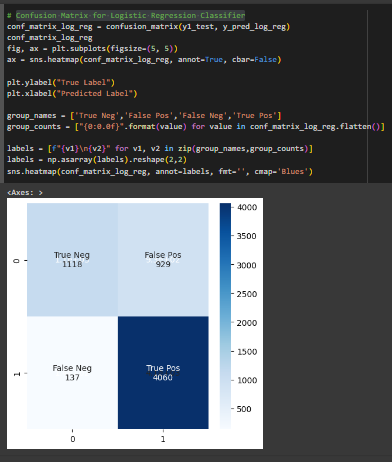


**Step 25**: Plotting the ROC AUC for Logistic Regression Classifier and plotting the Confusion Matrix for Logistic Regression Classifier

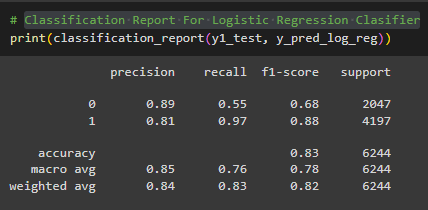


● **Logistic Regression**

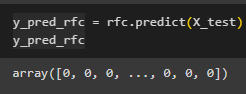
● **Rating between 3-4**



**Step 26**: Classification Report For Logistic Regression Classifier



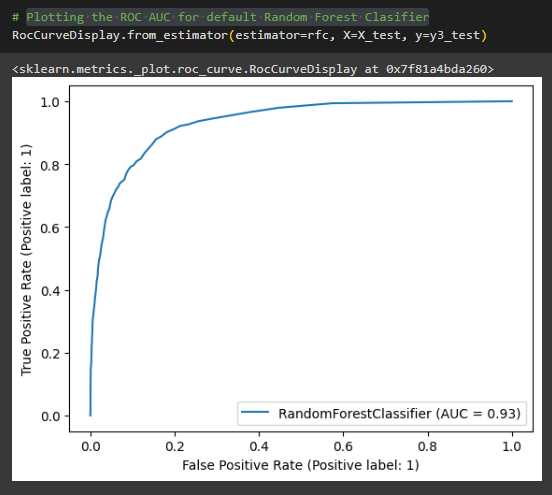
**Step 2**7: predicts the target variable values (or labels) for the test dataset (X\_test) using a trained Random Forest Classifier (rfc).



**Step 28**: Plotting the ROC AUC for default Random Forest Classifier

● **Random Forest**

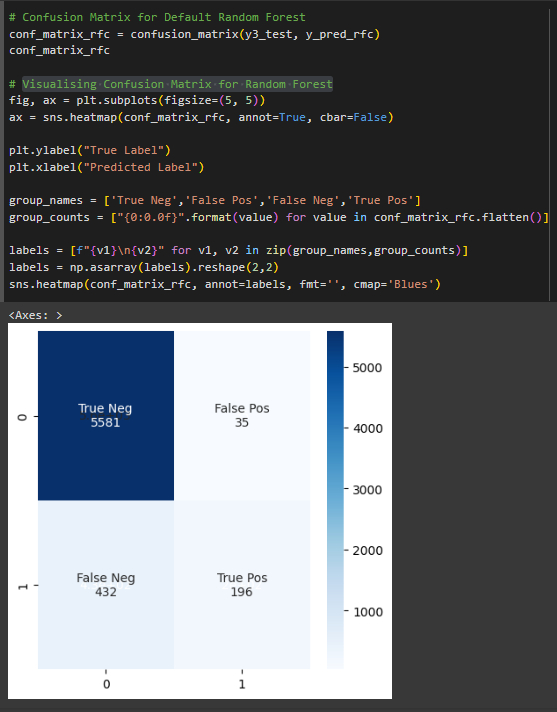
● **Rating above 4**



● **Random Forest**

● **Rating less than 3**

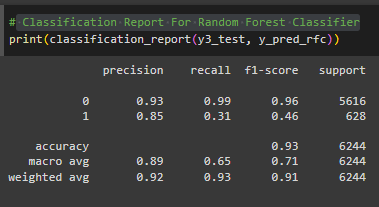
**Step 29**: Confusion Matrix for Default Random Forest and Visualising Confusion Matrix for Random Forest



● **Random Forest**

● **Rating between 3-4**

**Step 30**: Classification Report For Random Forest Classifier



**Future Scope:**

As with any analysis, it's essential to recognize that the outcomes and scores provided are subject to change as new data becomes available and methodologies evolve. Therefore, it's imperative to approach future analysis with an understanding that the conclusions drawn today may not necessarily hold true tomorrow. As our understanding deepens and datasets expand, different results might emerge, leading to adjustments in scores and interpretations. This underscores the dynamic nature of data analysis and the importance of continuous reassessment and refinement to ensure accuracy and relevance in decision-making processes.

In order to improve the model predictions and understanding data at a granular level we can add the unsupervised learning as they might produce different set of results and predictions.

Also we are planning to create an application using web development techniques such as django and flask to automate end to end process and apply our learnings at an industrial level.

**Conclusion:**

**Conclusions for Rating Above 4:**

* RandomForestClassifier and LogisticRegression perform similarly and are the best-performing models with scores just above 0.925, indicating a high level of accuracy in correctly classifying wines rated above 4.
* Knn (K-Nearest Neighbors) and SVC (Support Vector Classifier) are slightly less accurate but still perform well with scores just above 0.912 and 0.913 respectively.
* Decision Tree has the lowest accuracy among the models for this category, with a score of 0.903, which is still relatively high.

**Conclusions for Rating Between 3-4:**

* RandomForestClassifier leads with the highest accuracy of 0.836, suggesting that it is the most reliable model for predicting wines within this intermediate rating range.
* LogisticRegression and SVC both exhibit a good performance, with accuracy scores of 0.82, indicating they are equally competent for this category.
* Knn is slightly behind with an accuracy of 0.81.
* Decision Tree performs significantly worse in this category with an accuracy of 0.758, which might suggest overfitting on this more nuanced rating band or a lack of complexity to capture the variation within this range.

**Conclusions for Rating Less Than 3:**

* SVC performs the best in this category with an accuracy of 0.906, closely followed by RandomForestClassifier with 0.905 and LogisticRegression with 0.903, indicating all three models are highly effective at identifying wines rated below 3.
* Knn has a slightly lower accuracy of 0.89 but still performs quite well.
* Decision Tree again has the lowest accuracy with a score of 0.83, which may suggest its inability to generalize well to this lower rating class compared to the other models.

**Overall Conclusion:**

* RandomForestClassifier consistently performs well across all rating classes, making it a reliable choice for this prediction task.
* LogisticRegression and SVC also show strong and consistent performance, particularly for the extreme rating categories (above 4 and less than 3).
* Knn is a solid performer but slightly less accurate than the best models in each category.
* Decision Tree seems to struggle relative to the other models, especially in the intermediate rating range, which could indicate a limitation in capturing more complex patterns without overfitting.

**Achievement after formulating and Data Wrangling**

**Formulating the Decision tree by wrangling column number of votes increased accuracy from 60 % to 90 % decision tree as shown below**

