Feasting on Data: Unveiling the Secret Recipe of Restaurant Ratings on Zomato

Ganeshan Malhotra
UC San Diego
A59019461
gmalhotra@ucsd.edu

Mayank Umeshprasad Sharma
UC San Diego
A59019439
musharma@ucsd.edu

1 Abstract

This study utilizes the **Zomato Dataset**[1] to analyze the factors affecting restaurant ratings in Bengaluru, India's vibrant culinary hub. We focus on a variety of features including cuisine diversity, pricing, and location to understand their impact on customer ratings. Through meticulous feature engineering and analysis, we develop a predictive model to estimate a restaurant's average rating, employing techniques like Linear Regression, Decision Trees, and Random Forests for a comparative performance study.

Our model not only forecasts ratings but also offers insights into Bengaluru's dining preferences, serving as a valuable tool for stakeholders in the restaurant industry. The findings of this study underline the significance of multifaceted data analysis in predicting consumer behavior and guiding business strategies in the gastronomy sector. This research exemplifies the integration of statistical learning and real-world market dynamics, showcasing the potential of data analytics in the hospitality domain. The code is attached below and can also be found at the GitHub[2] link.

2 Significance

This project holds significance for the hospitality industry. By predicting ratings on Zomato, we aim to bridge the gap between customer expectations and restaurant services. The implications of this study are twofold: Firstly, it provides restaurateurs with data-driven insights into consumer preferences, aiding in service improvement and targeted marketing. Secondly, it offers a methodological framework for similar predictive analyses in other service-oriented domains. This study, therefore, contributes not only to the academic field of predictive modeling but also offers practical applications in enhancing customer satisfaction and business strategies in the fast-evolving food industry.

3 Dataset

The dataset contains records of unique 51717 restaurants. The following features are present in the dataset. The *url* shows the corresponding web-url page of the restaurant. The *address* field shows the address where the restaurant is located. The *name* field represents the name of the restaurant. The *online order* is a binary column which shows if that restaurant supports the serivce of online ordering whereas the *book table* is also a binary column which shows the if that restaurant supports the option of booking or reserving a table. The *votes* shows the number of votes/ratings that the restaurant has received. The *cuisines* shows the types of cuisines serverd in that restaurant. The *approx cost* column contains the approximate cost for two people in that restaurant. The *reviews list* column contains the pairs of ratings and reviews for the restaurant. The *menu item* shows the list of menu items served in the restaurant.

3.1 Exploratory Data Analysis

In this section, we present the exploratory analysis of the dataset. Figure 1 shows the distribution of the number of restaurants v/s their rating. This distribution roughly resembles a bell-curve.

Figure 2 shows the distribution of the top cuisines in the city. It can be clearly seen that the top cuisine is *North Indian* followed by *Chinese* and *South Indian*.

Figure 3 shows the pie-chart representing the proportion of restaurants accepting online orders. There are a total of 30444 restaurants which accept online orders while a total of 21273 do not accept online orders.

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

[2] https://github.com/mayankmssharma/ECE-225-Project/tree/main

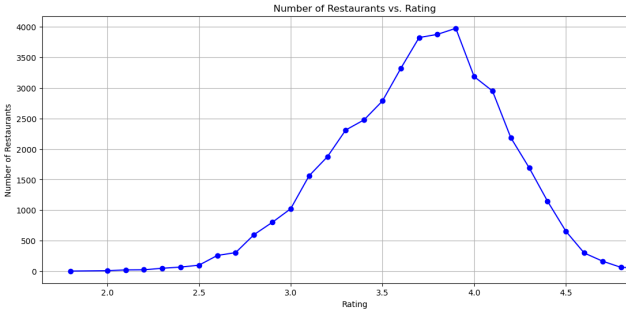


Figure 1: Graph showing the distribution of number of restaurants v/s their rating. It roughly follows a normal distribution.

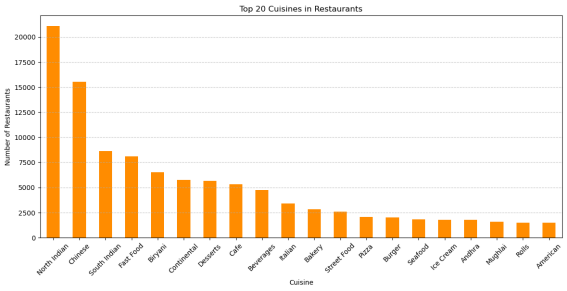


Figure 2: Figure showcasing the top cuisines in Bengaluru

Figure 4 shows the distribution of the total number of votes grouped by the locality of the restaurants in the city. It can be clearly observed from the graph that some localities tend to receive more votes as compared to others.

Figure 5 shows two distributions for ratings v/s the availability of the reserving table service. When the restaurant offers this service, the distribution is skewed towards the right showcasing that the average rating tends to be higher in this case. Figure 6 shows the cost comparisons when the option of online order is available as compared to when it is not. When the option of online order is available the median cost is 400, the third quartile being at 550 whereas when the option of online order is not available the median cost is 300, the third quartile being 450.

Figure 7 shows two distributions for approximate cost (for two people) v/s the availability of reserving a table service. When the restaurant offers this service, the distribution is skewed towards the right showcasing that the average cost for two people tends to be higher in this case.

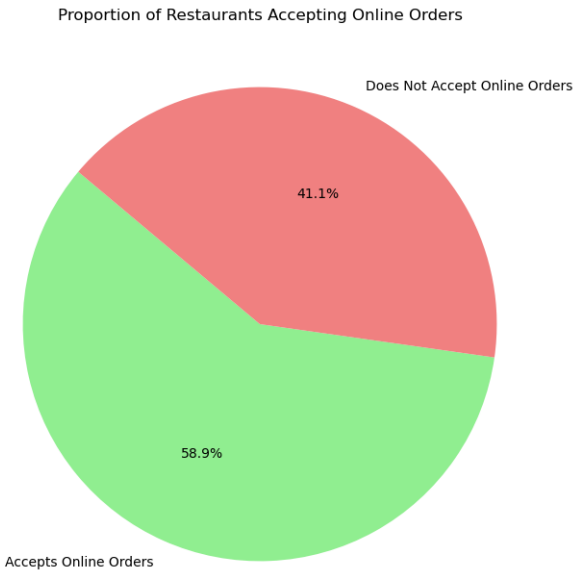


Figure 3: Graph showing the proportion of restaurants accepting online orders

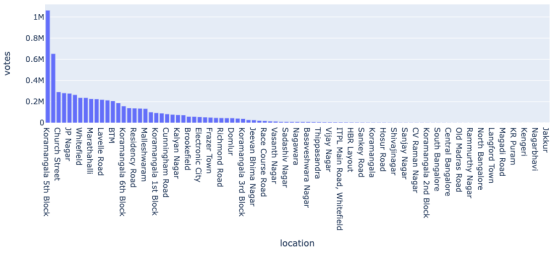


Figure 4: Graph showing the number of votes available for each location in the city. Some areas receive more votes as compared to others.

4 Hypothesis Testing and Measuring the fit of the Distributions

Consider the distributions shown in Figure 5. The distributions in both the scenarios resemble gaussian distributions.In this section, we aim to test this hypothesis. The mean and the standard deviation of the samples when Book Table service is available are 4.143 and 0.301 respectively, whereas the mean and the standard deviation of the samples when Book Table service is not available are 3.621 and 0.413 respectively. Now, using T TEST, we measure the fit of the distributions on the data. We consider the following null hypothesis:

- $H_{0,a}$: The mean of the distribution when book table service is available is 4.14
- $H_{0,b}$: The mean of the distribution when book table service is available is 3.6214

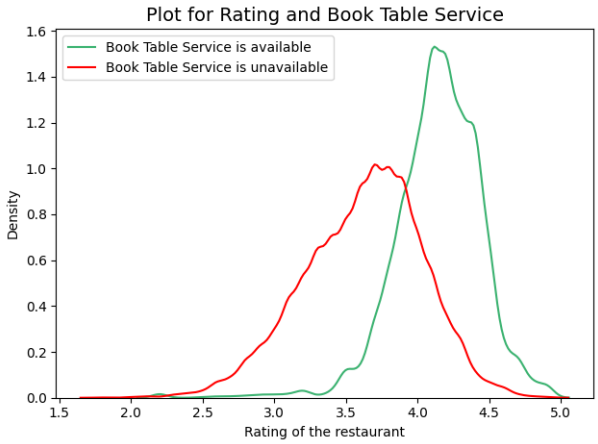


Figure 5: Graph showing the distribution of Ratings v/s the presence of book table service. It can be clearly seen that when this service is available the average rating is higher.

Figure 6: Graph showing the cost comparison when the option of online order is available as compared to when it is not

The observations are given in Table 1. The test statistic is 0.9108 and the p-value is 0.3624 whereas when the book table service is not available the test statistic is 0.0318 and the p-value is 0.9746. Since, the p-value is greater than 0.05 in both the cases, we can retain the null hypothesis that the mean of these distributions is 4.14 and 3.6214 respectively. However, since the p-value is greater in the second case, we can have much more statistical confidence when retaining the null hypothesis. This also means that we have better fit for the distribution as shown in Figure 8.

Now, we also test and the *power* of the T TEST conducted above. Power is defined as the probability test will correctly reject a false null hypothesis. In other words, it is the probability of avoiding a Type II error, which occurs when you fail to reject

Book Table Service	Test Statistic	P Value
Available	0.9108	0.3624
Not Available	0.0318	0.9746

Table 1: Table showing the p-value and the test statistic for conducting t test on ratings of restaurant in two scenarios a. when the book table service is availabe and b. when this service is unavailable

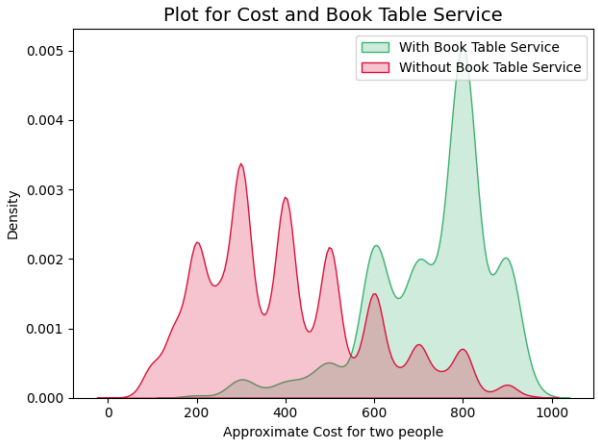


Figure 7: Graph showing the distribution of Approximate cost for two people v/s the presence of book table service. It can be clearly seen that when this service is available the average cost tends to be higher.

a null hypothesis that is actually false.

5 Correlation Analysis Using Kendall's Tau Coefficient

In our research project focused on predicting ratings on Zomato data, we analyze the correlation between various features to understand their interrelationships better. After excluding non-relevant features such as URL, PHONE, and DISH LIKED, we focus on the remaining 12 significant features. To quantify the correlation, we employ Kendall's Tau(Kendall, 1938) correlation coefficient, a non-parametric measure that evaluates the strength and direction of association between two variables.

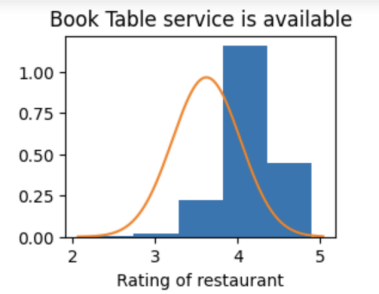
Kendall's Tau is calculated based on the concordance and discordance of pairs in the dataset. It is defined as:

$$\tau = \frac{C - D}{C + D} \tag{1}$$

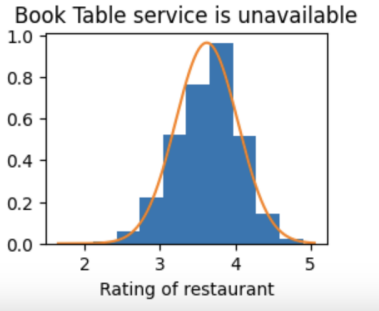
where C represents the number of concordant pairs, and D denotes the number of discordant pairs. A pair is considered concordant if the order of the observations in one variable corresponds to the order in the other variable, and discordant if it is opposite.

We observed notable correlations in our dataset, such as a strong correlation between the restaurant's address and its name. This correlation is intuitive, suggesting that restaurant owners often choose names that appeal to the locality's audience.

To visually represent these correlations, we constructed a heatmap, as shown in Figure 9, which provides a comprehensive overview of how each



(a)



(b)

Figure 8: Figures showing the differences in the fit of distributions for the two cases

feature is related to the others in the context of Zomato data.

6 Using BERT model to Predict Ratings using Reviews

In this section, we use reviews to train models to predict ratings of a restaurant. Previous works like (Cheng et al., 2018), (Lei et al., 2016) and (Ahmed and Ghabayen, 2022) have used reviews to predict ratings in various domains. For each restaurant, we have a bunch of reviews along with a rating attached to it. The rating ranges from 1 stars (minimum) to 5 stars (maximum). We used BERT Model from huggingface [3]along with a linear layer to classify the ratings.

6.1 BERT

BERT, or Bidirectional Encoder Representations from Transformers (Devlin et al., 2018), is a groundbreaking method in the field of natural language processing (NLP). Developed by Google, BERT revolutionized how machines understand human language. Its core innovation lies in its ability to process words in relation to all the other words in a sentence, rather than one-by-one in order. This

[3] https://huggingface.co/bert-base-uncased

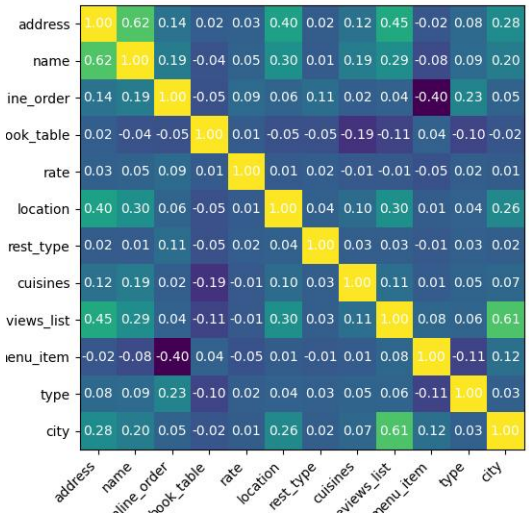


Figure 9: Heatmap showing the correlation between different features present in the dataset

is achieved through the use of the transformer architecture, which employs attention mechanisms to understand the context of a word within its surrounding text.

One of the key mathematical foundations of BERT is the attention mechanism, specifically the scaled dot-product attention. The attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. The equation for the scaled dot-product attention is:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2)$$

In this equation, Q, K, and V represent the matrices for queries, keys, and values, respectively. d_k denotes the dimension of the key vectors. The scaling factor $\sqrt{d_k}$ prevents the dot products from becoming excessively large, thus averting issues related to gradient vanishing during training.

In the context of predicting ratings on Zomato data, BERT can be particularly effective for understanding the nuances and context of user reviews. It takes a lot of time to train BERT models from scratch. Hence, we use the pretrained *bert-base-uncased*. Example of pre-training and fine-tuning can be seen in figure 10 The model was trained

on 9000 reviews and evaluated on 1000 reviews. The training and validation loss and accuracy are mentioned below in the table 2.

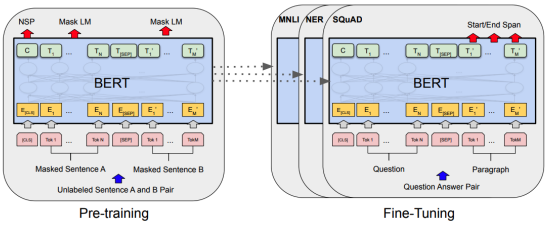


Figure 10: BERT model architecture

Metric	Training	Validation
Loss	0.16756	0.65657
Accuracy	94.811%	85.4%

Table 2: Training and Validation Metrics for BERT Model

7 Calculating Rating using Numeric Features

In this section, we aim to calculate the rating of a restaurant using only the numeric features present in the data. The features include ONLINE ORDER, BOOK TABLE, LOCATION, REST TYPE, LOCATION, CUISINES, REVIEWS LIST, MENU ITEMS, CITY. We experimented using the following models - Linear Regression, Decision Tree Regression, Random Forest Regression and Gradient Boosting Regression. We use sklearn library (Pedregosa et al., 2011) to perform these experiments.

7.1 Linear Regression

Linear Regression[4] (Montgomery et al., 2021) is one of the simplest and most widely used statistical techniques for predictive modeling. It assumes a linear relationship between the dependent variable and one or more independent variables. The general form of a linear regression model with n independent variables is given by:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon \quad (3)$$

where y is the dependent variable, $x_1, x_2, \ldots, x_n$ are the independent variables, β_0 is the y-intercept, $\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients of the independent variables, and ϵ is the error term. This model aims to find the best-fitting linear relationship between the independent and dependent variables.

7.2 Decision Tree Regression

Decision Tree Regression[5] operates by splitting the data into different regions based on the feature values. In each region, the model makes a prediction, usually the mean of the target values in that region. The decision to split at each node is based on reducing the Mean Squared Error (MSE). The MSE for a node is calculated as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2 \quad (4)$$

where y_i are the actual values, $\hat{y}$ is the predicted value for the node, and n is the number of samples in the node.

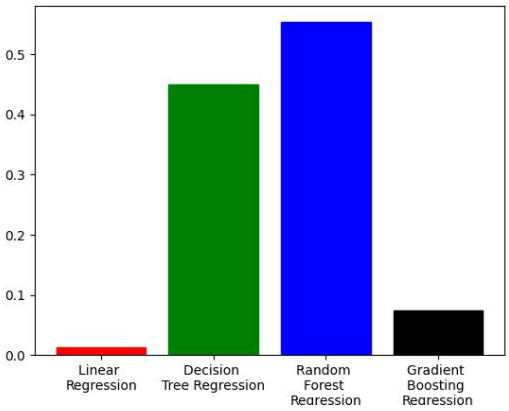


Figure 11: Figure showing the R2 score for different ML models

7.3 Random Forest Regression

Random Forest Regression [6]builds upon Decision Trees. It creates an ensemble of decision trees, each trained on a random subset of the data and features. The final prediction is the average of the predictions from all trees. The prediction of the random forest regressor is given by:

$$y = \frac{1}{B}\sum_{b=1}^{B} T_b(\mathbf{x}) \quad (5)$$

where T_b represents individual decision trees and B is the number of trees in the forest.

[4] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

[5] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html

[6] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

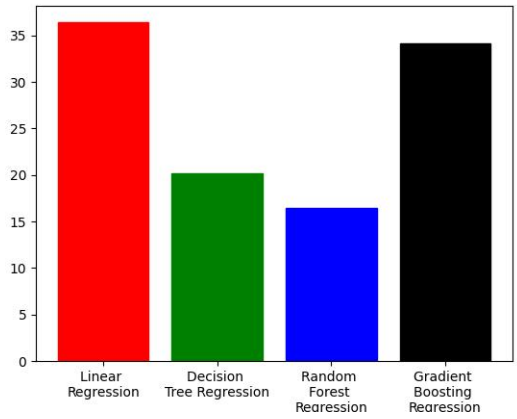


Figure 12: Figure showing the MSE score for different ML models

7.4 Gradient Boosting Regression

Gradient Boosting Regression [7] involves sequentially adding decision trees, where each new tree corrects the errors made by the previous ones. The model's prediction is a weighted sum of the predictions of these trees. The formulation of the Gradient Boosting Regressor can be described as:

$$y = \sum_{b=1}^{B} \alpha_b T_b(\mathbf{x}) \tag{6}$$

where α_b are the weights assigned to the trees T_b, and B is the number of trees.

7.5 Comparative Analysis for Machine Learning Models

Through our experiments, we found that the Random Forest model performs the best with a mean squared error (MSE) of 16.45% and R2 score of 0.553. This can be attributed to various factors. Random Forests are generally better at handling non-linearity as compared to Linear Regression which, by default, assumes a linear relationship between the features and the target variable. Decision Trees and Gradient Boosting methods are often prone to overfitting on the training data and therefore Random Forests tend to perform better. Also, Random Forests are more robust to outliers as compared to Gradient Boosting methods in the data thereby giving better performance.

[7] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html

8 Conclusion

In this paper, we presented an extensive exploratory data analysis on the Zomato data which contains information about various restaurants in the city of Bengaluru. Through rigorous data analysis and feature engineering, we have tried to provide insights into factors influencing ratings. For reviews data we have used the BERT model which can classify the ratings for a restaurant. On the other hand for numeric data, we project the problem as a regression problem. We have used Machine Learning models like Linear Regression, Decision Trees, Random Forests and Gradient Boosting regression. The BERT model gave an accuracy of 85.4% trained over 5 epochs while among the Machine Machine Learning models the Random Forest model performed the best giving a mean squared error of 16.45%. We expect much higher accuracy with BERT if we train it longer as there validation accuracy was consistently decreasing.

For future work, we recommend exploring deeper neural network architectures and more advanced natural language processing techniques like Large Language Models to better capture the subtleties in customer reviews. Additionally, integrating numeric data with the reviews data may also prove better in terms of improving the accuracy of models on downstream tasks. Also, geospatial data for a more detailed analysis of location-based trends and experimenting with real-time data to predict dynamic rating changes could further enhance the model's applicability. This research lays a foundation for more nuanced and comprehensive predictive analytics in the restaurant industry.

References

Basem H Ahmed and Ayman S Ghabayen. 2022. Review rating prediction framework using deep learning. *Journal of Ambient Intelligence and Humanized Computing*, 13(7):3423–3432.

Zhiyong Cheng, Ying Ding, Lei Zhu, and Mohan Kankanhalli. 2018. Aspect-aware latent factor model: Rating prediction with ratings and reviews. In *Proceedings of the 2018 world wide web conference*, pages 639–648.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.

Xiaojiang Lei, Xueming Qian, and Guoshuai Zhao. 2016. Rating prediction based on social sentiment from textual reviews. *IEEE transactions on multimedia*, 18(9):1910–1921.

Douglas C Montgomery, Elizabeth A Peck, and G Geoffrey Vining. 2021. *Introduction to linear regression analysis*. John Wiley & Sons.

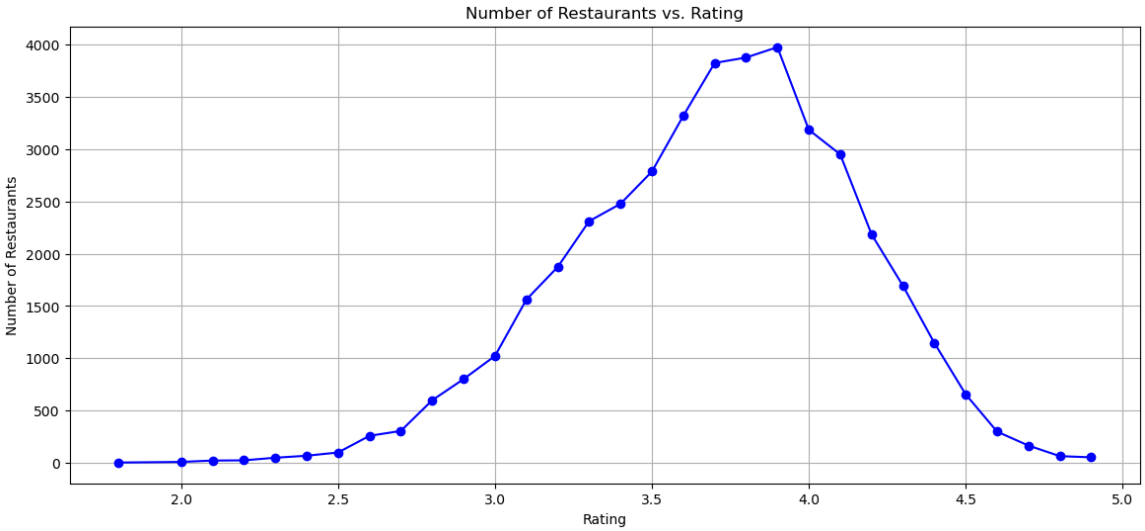
F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

In [ ]:
```
# from google.colab import drive
# drive.mount('/content/drive')
```

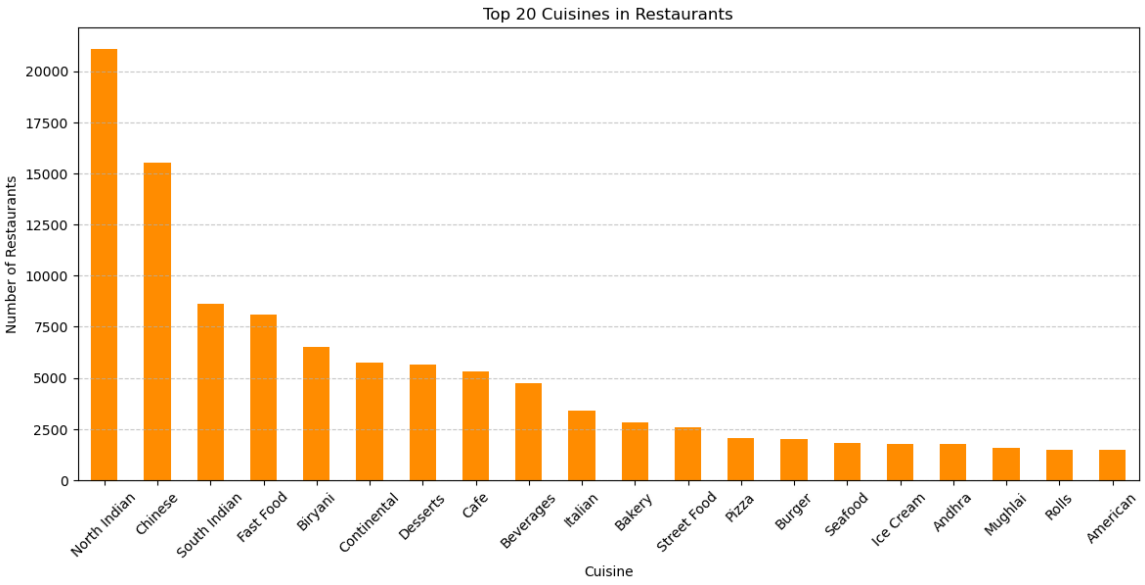
In [ ]:
```
import pandas as pd
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt
import plotly.express as px
from scipy.stats import ttest_1samp

# df = pd.read_csv('/content/drive/MyDrive/ECE 225 Project/zomato.csv')
df = pd.read_csv('zomato.csv')
```

EDA

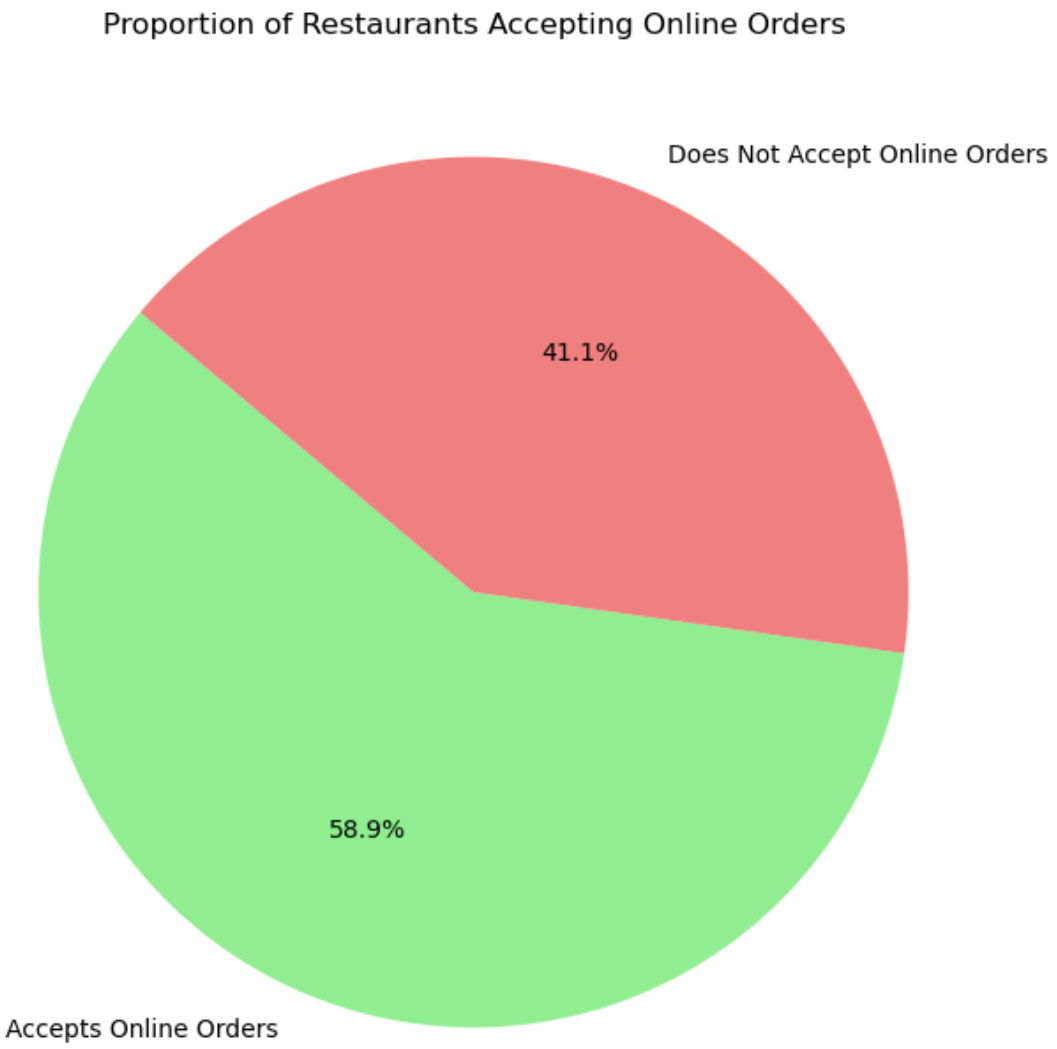
In [ ]:
```
df['rate'] = df['rate'].dropna().apply(lambda x : float(x.split('/')[0])
# Group the data by ratings and count the number of restaurants for each
ratings_group = df.groupby('rate').size()

# Creating the plot
plt.figure(figsize=(14, 6))
ratings_group.plot(kind='line', marker='o', color='blue')
plt.title('Number of Restaurants vs. Rating')
plt.xlabel('Rating')
plt.ylabel('Number of Restaurants')
plt.grid(True)
plt.show()
```



In [ ]:
```
cuisine_counts = df['cuisines'].dropna().str.split(', ').explode().value_

# Creating the plot for the number of restaurants vs cuisine
plt.figure(figsize=(14, 6))
cuisine_counts.head(20).plot(kind='bar', color='darkorange')  # Displayin
plt.title('Top 20 Cuisines in Restaurants')
plt.xlabel('Cuisine')
plt.ylabel('Number of Restaurants')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



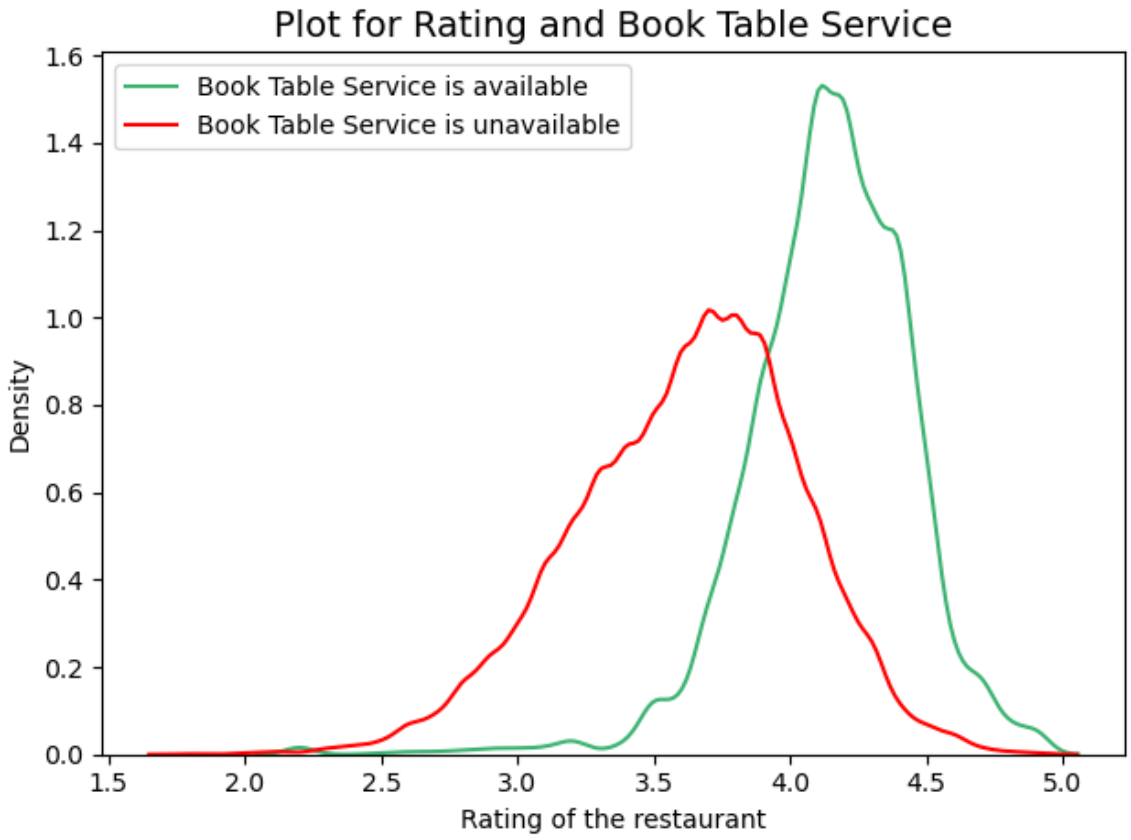
In [ ]:
```
online_order_proportions = df['online_order'].value_counts(normalize=True

# Creating a pie chart
plt.figure(figsize=(8, 8))
online_order_proportions.plot(kind='pie', labels=['Accepts Online Orders'
plt.title('Proportion of Restaurants Accepting Online Orders')
plt.ylabel('')  # Hiding the y-label for clarity in a pie chart
plt.show()
```



In [ ]:
```
df = df.rename(
    columns={
        "approx_cost(for two people)": "cost",
        "listed_in(type)": "type",
        "listed_in(city)": "city",
    }
)
df["cost"] = df["cost"].astype(str)
df["cost"] = df["cost"].apply(lambda x: x.replace(",", "."))
df["cost"] = df["cost"].astype(float)

fig = px.box(
    df,
    x="online_order",
    y="cost",
    color="online_order",
)

fig.update_layout(
    title="Cost comparison for Online order",
    titlefont={"size": 30},
    template="simple_white",
)
fig.show()
```

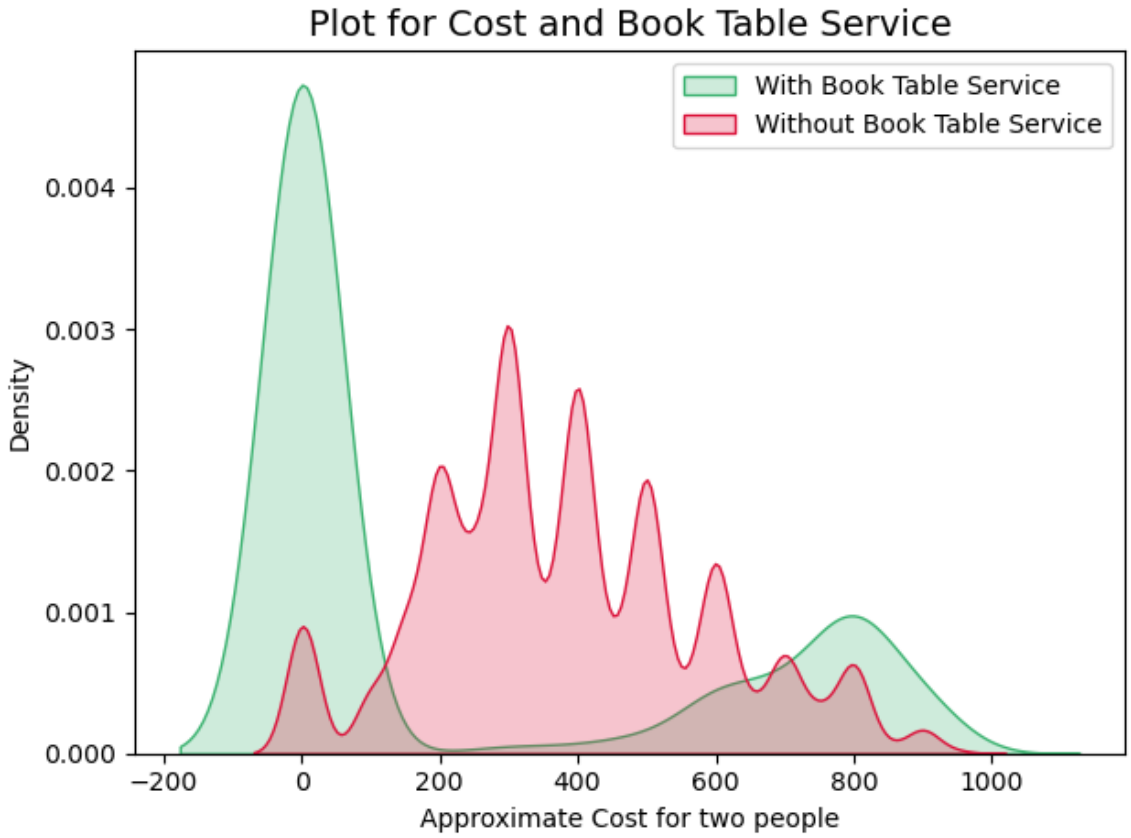
In [ ]:
```
px.bar(
    df,
    x="location",
    y="votes",
    labels={"value": "Sum of Values"},
)
```

In [ ]:
```
import seaborn as sns

df["rate_num"] = df["rate"].astype(str).apply(lambda x: x.split("/")[0])
df["rate_num"] = pd.to_numeric(df["rate_num"], errors="coerce")

fig, ax = plt.subplots()

sns.kdeplot(
    df.query('rate_num > 0 & book_table == "Yes"')["rate_num"],
    color="mediumseagreen",
    label="Book Table Service is available",
)
sns.kdeplot(
    df.query('rate_num > 0 & book_table == "No"')["rate_num"],
    color="red",
    label="Book Table Service is unavailable",
)
ax.set_title("Plot for Rating and Book Table Service", size=14)
ax.set_xlabel("Rating of the restaurant")
ax.legend()
plt.tight_layout()
```



In [ ]:
```
fig, ax = plt.subplots()
sns.kdeplot(
```

```
    df.query('cost > 0 & book_table == "Yes"')[
        "cost"
    ],
    color="mediumseagreen",
    shade=True,
    label="With Book Table Service",
)
sns.kdeplot(
    df.query('cost > 0 & book_table == "No"')[
        "cost"
    ],
    color="crimson",
    shade=True,
    label="Without Book Table Service",
)

ax.set_title("Plot for Cost and Book Table Service", size=14)
ax.set_xlabel("Approximate Cost for two people")
ax.legend()
plt.tight_layout()
```



Get the mean and standard deviation

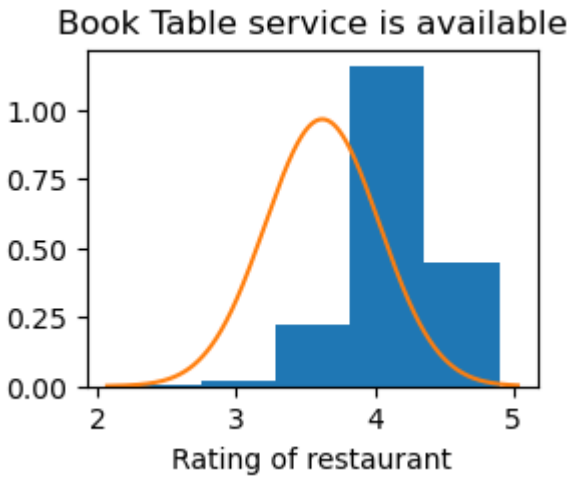
In [ ]:
```
df["rate_num"] = df["rate"].astype(str).apply(lambda x: x.split("/")[0])
df["rate_num"] = pd.to_numeric(df["rate_num"], errors="coerce")
mean, std = norm.fit(df.query('rate_num > 0 & book_table =="Yes"')['rate_
print(f"Mean and Std when Book Table is available: {mean}, {std}")

mean, std = norm.fit(df.query('rate_num > 0 & book_table =="No"')['rate_n
print(f"Mean and Std when Book Table is unavailable: {mean}, {std}")
```

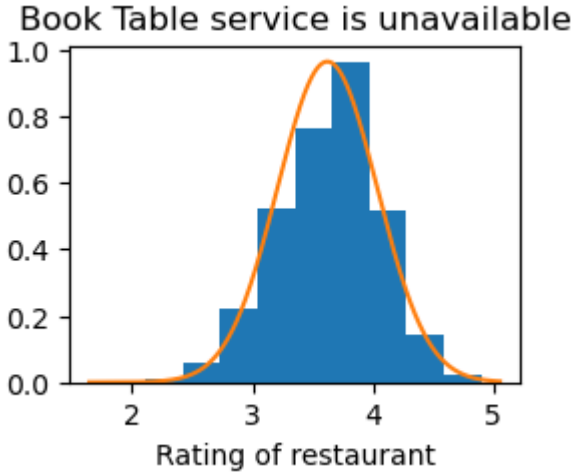
```
Mean and Std when Book Table is available: 4.143464467005076, 0.30197392
482709057
Mean and Std when Book Table is unavailable: 3.621469981052572, 0.413711
8437005144
```

Perform T test and plot the fit

In [ ]:
```
def plot_and_perform_ttest_rating_book_table():

    plt.subplot(2,2,1)
    plt.hist(df.query('rate_num > 0 & book_table =="Yes"')['rate_num'], d
    xmin, xmax = plt.xlim()
    x = np.linspace(xmin, xmax, 1000)
    y = norm.pdf(x, mean, std)
    plt.plot(x, y, label='pdf')
    plt.xlabel('Rating of restaurant')
    plt.title('Book Table service is available')
    plt.show()

    plt.subplot(2,2,2)
    plt.hist(df.query('rate_num > 0 & book_table =="No"')['rate_num'], de
    xmin, xmax = plt.xlim()
    x = np.linspace(xmin, xmax, 1000)
    y = norm.pdf(x, mean, std)
    plt.plot(x, y, label='pdf')
    plt.xlabel('Rating of restaurant')
    plt.title('Book Table service is unavailable')

    plt.show()


    print(ttest_1samp(a = df.query('rate_num > 0 & book_table =="Yes"')['
    print(ttest_1samp(a = df.query('rate_num > 0 & book_table =="No"')['r
```

In [ ]:
```
plot_and_perform_ttest_rating_book_table()
```






```
Ttest_1sampResult(statistic=0.9108369481303978, pvalue=0.362416122662603
5)
Ttest_1sampResult(statistic=0.031808167119425045, pvalue=0.9746252130387
685)
```

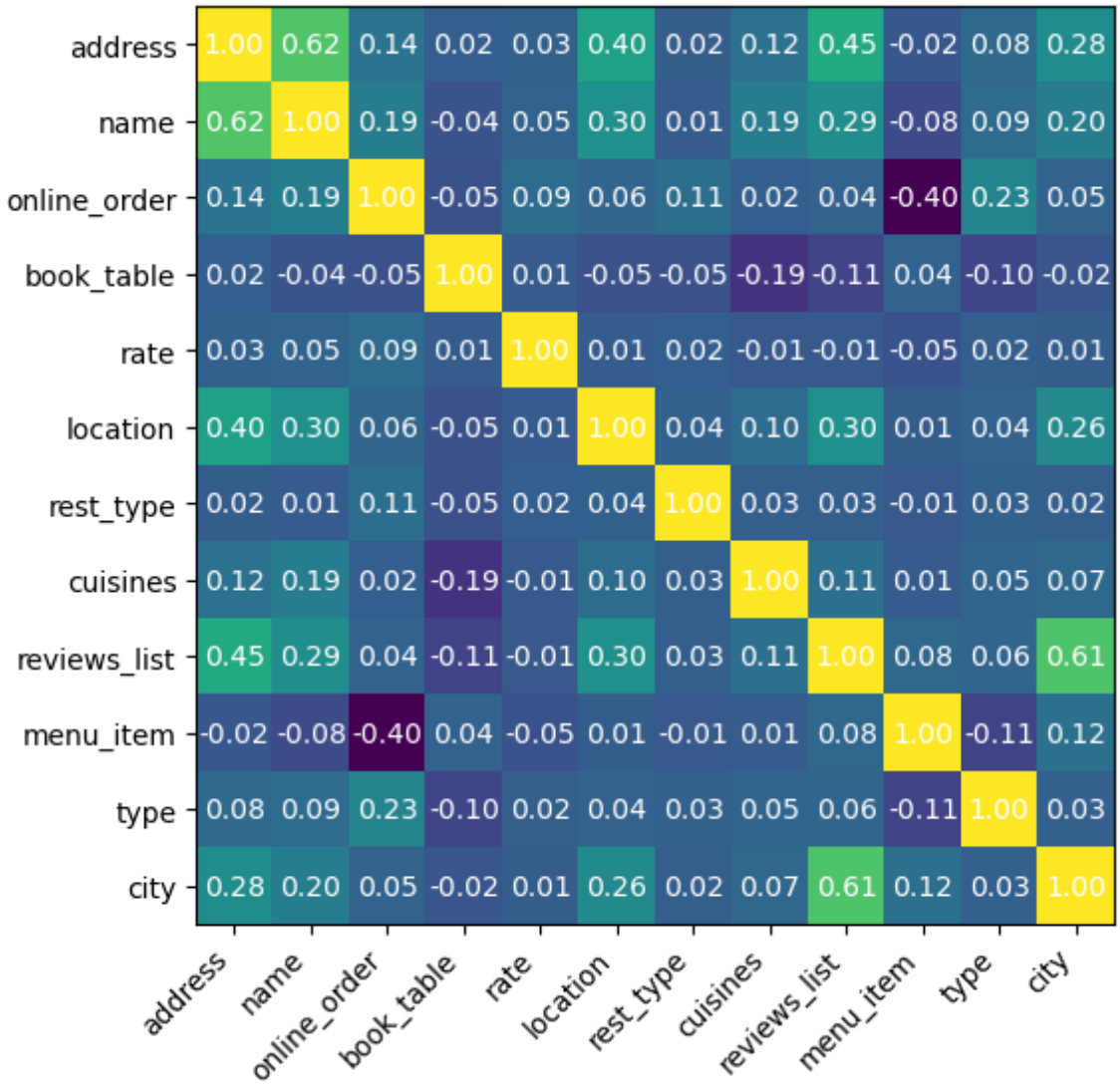
Create encoded data

In [ ]:
```
df = pd.read_csv('zomato.csv')
def create_encoded_df(input_df):
    input_df.drop(['url','phone','dish_liked'],axis=1,inplace=True)
    input_df.drop_duplicates(inplace=True)
    input_df.dropna(how='any',inplace=True)
    input_df = input_df.rename(columns={'approx_cost(for two people)':'co
    input_df['cost']=input_df['cost'].astype(str)
    input_df['cost'] = input_df['cost'].apply(lambda x: x.replace(',','.'
    input_df['cost'] = input_df['cost'].astype(float)

    input_df = input_df.loc[input_df.rate !='NEW']
    input_df = input_df.loc[input_df.rate !='-'].reset_index(drop=True)
    remove_slash = lambda x: x.replace('/5', '') if type(x) == str else x
    input_df.rate = input_df.rate.apply(remove_slash).str.strip().astype(
    input_df['rate'].head()
    input_df.name = input_df.name.apply(lambda x:x.title())
    input_df.online_order.replace(('Yes','No'),(True, False),inplace=True
    input_df.book_table.replace(('Yes','No'),(True, False),inplace=True)

    output_df = pd.DataFrame()
    for column in input_df.columns[~input_df.columns.isin(['rate_num','co
        output_df[column] = pd.factorize(input_df[column])[0]
    return output_df

prediction_df = create_encoded_df(df.copy())
```

Get correlation

In [ ]:
```
def get_correlation(df):
    corr = df.corr(method="kendall")
    plt.figure(figsize=(6, 6))
    plt.imshow(corr)
    corr_data = corr.to_numpy()
    plt.xticks(np.arange(len(corr)), labels=list(corr.columns))
    plt.yticks(np.arange(len(corr)), labels=list(corr.columns))
    plt.xticks(rotation=45, ha='right')
```

```
    for i in range(12):
        for j in range(12):
            plt.annotate('{0:.2f}'.format(corr_data[i][j]), xy=(j, i),
                         ha='center', va='center', color='white', annotat
    plt.savefig('research_corr.jpg')
    plt.show()
get_correlation(prediction_df)
```



Train and test ML models

In [ ]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import  GradientBoostingRegressor
from sklearn.metrics import r2_score, mean_squared_error

def get_train_test_split(input_df):

    x = input_df.iloc[:,[2,3,5,6,7,8,9,11]]
    y = input_df['rate']
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.2,rando
    return {'x_train': x_train, 'y_train':y_train, 'x_test':x_test, 'y_te

def fit_model(model,data):
```

```
    model.fit(data['x_train'], data['y_train'])
    y_predicted = model.predict(data['x_test'])
    r2 = r2_score(data['y_test'],y_predicted)
    mse = mean_squared_error(data['y_test'],y_predicted)

    print(f"R2 Score of {model.__repr__()} is {r2}")
    print(f"MSE Score of {model.__repr__()} is {mse}")
    return r2, mse

data = get_train_test_split(prediction_df)



r2 = []
mse = []

LRmodel=LinearRegression()
metrics = fit_model(LRmodel, data)
r2.append(metrics[0])
mse.append(metrics[1])

DTmodel=DecisionTreeRegressor(min_samples_leaf=.0001)
metrics = fit_model(DTmodel, data)
r2.append(metrics[0])
mse.append(metrics[1])

RFmodel=RandomForestRegressor(n_estimators=100,random_state=42,min_sample
metrics = fit_model(RFmodel, data)
r2.append(metrics[0])
mse.append(metrics[1])


GBmodel=GradientBoostingRegressor()
metrics = fit_model(GBmodel, data)
r2.append(metrics[0])
mse.append(metrics[1])
```

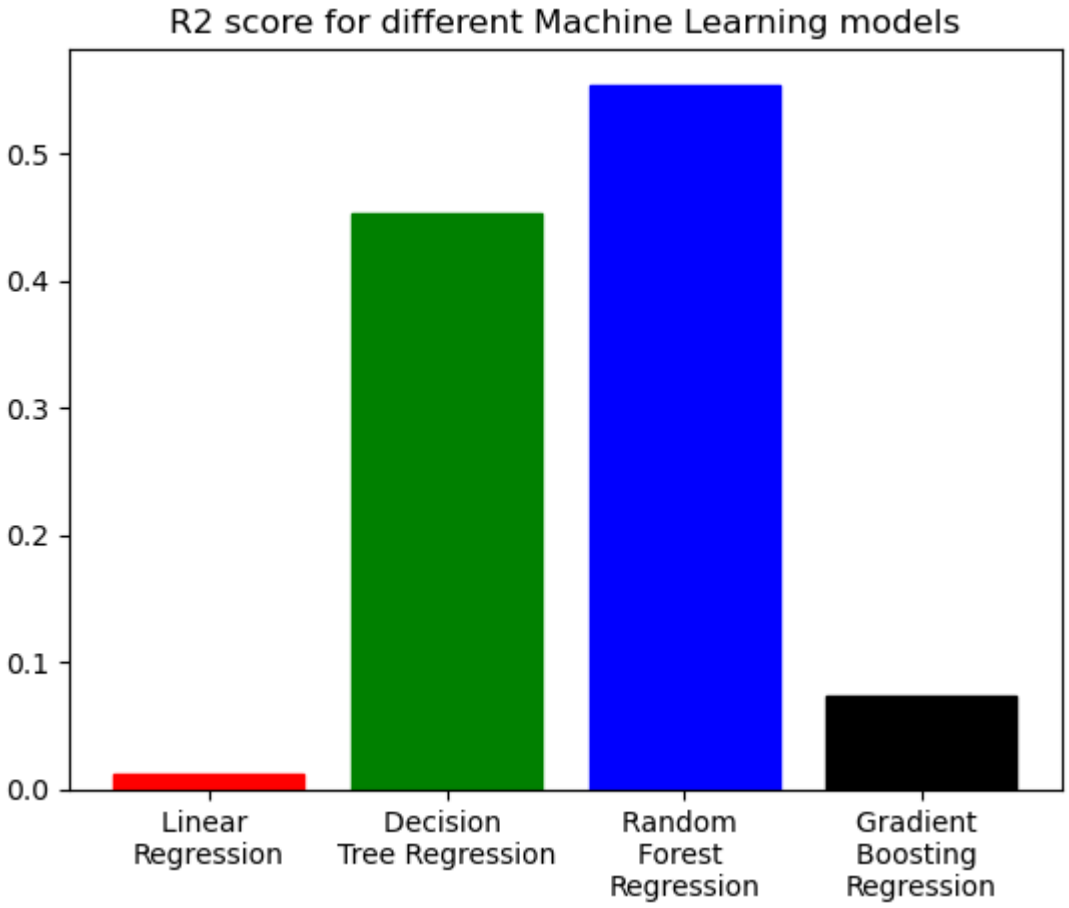
```
R2 Score of LinearRegression() is 0.012971694196955319
MSE Score of LinearRegression() is 36.39514558179139
R2 Score of DecisionTreeRegressor(min_samples_leaf=0.0001) is 0.45239938
8798723
MSE Score of DecisionTreeRegressor(min_samples_leaf=0.0001) is 20.191927
473785466
R2 Score of RandomForestRegressor(min_samples_leaf=0.0001, random_state=
42) is 0.5536198481290605
MSE Score of RandomForestRegressor(min_samples_leaf=0.0001, random_state
=42) is 16.45957924068572
R2 Score of GradientBoostingRegressor() is 0.07438268091564304
MSE Score of GradientBoostingRegressor() is 34.130710216759326
```

Plot results for ML models

In [ ]:
```
x = [f'Linear \nRegression', 'Decision \nTree Regression', 'Random \nFore
y = r2
my_colors = 'rgbkymc'

barlist = plt.bar(x, y)
barlist[0].set_color('r')
barlist[1].set_color('g')
```

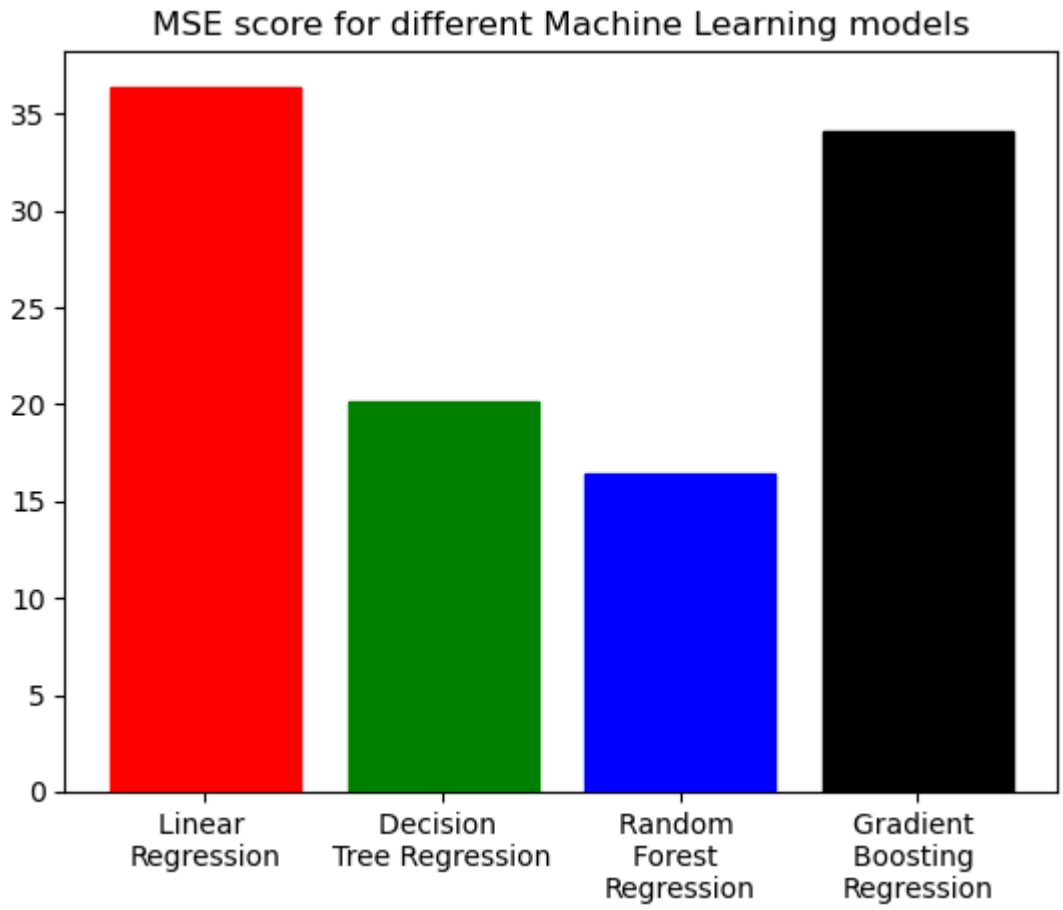
```
barlist[2].set_color('b')
barlist[3].set_color('k')
plt.savefig('R2Score.jpg')
plt.title('R2 score for different Machine Learning models')
```

Out[ ]: Text(0.5, 1.0, 'R2 score for different Machine Learning models')



In [ ]:
```
x = [f'Linear \nRegression', 'Decision \nTree Regression', 'Random \nFore
y = mse
my_colors = 'rgbkymc'

barlist = plt.bar(x, y)
barlist[0].set_color('r')
barlist[1].set_color('g')
barlist[2].set_color('b')
barlist[3].set_color('k')
plt.savefig('MseScore.jpg')
plt.title('MSE score for different Machine Learning models')
```

Out[ ]: Text(0.5, 1.0, 'MSE score for different Machine Learning models')


MSE score for different Machine Learning models
35
30
25
20
15
10
5
0
Linear Regression
Decision Tree Regression
Random Forest Regression
Gradient Boosting Regression

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [ ]:
```
import pandas as pd
df = pd.read_csv('/content/drive/MyDrive/ECE 225 Project/zomato.csv')
data = df
data.head()
```

Out[ ]:

	url	address	name	online_order	book
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	
3	https://www.zomato.com/bangalore/addhuri-udupi...	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	
4	https://www.zomato.com/bangalore/grand-village...	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	No	

In [ ]:
```
data.iloc[1]
```

Out[ ]:
```
url                                  https://www.zomato.com/bangalore/spice-el
ephan...
address                              2nd Floor, 80 Feet Road, Near Big Bazaar,
6th ...
name                                                                 Spice
Elephant
online_order
Yes
book_table
No
rate
4.1/5
votes
787
phone                                                                 080
41714161
location                                                             Bana
shankari
rest_type                                                           Casua
l Dining
dish_liked                           Momos, Lunch Buffet, Chocolate Nirvana, T
hai G...
cuisines                                                Chinese, North Indi
an, Thai
approx_cost(for two people)
800
reviews_list                         [('Rated 4.0', 'RATED\n  Had been here fo
r din...
menu_item
[]
listed_in(type)
Buffet
listed_in(city)                                                      Bana
shankari
Name: 1, dtype: object
```

In [ ]:
```
from tqdm import tqdm
all_ratings = []
flag = False
for name,ratings in tqdm(zip(df['name'],df['reviews_list'])):
    ratings = eval(ratings)
    for score, doc in ratings:

        if score:
            score = score.strip("Rated").strip()
            doc = doc.strip('RATED').strip()
            score = float(score)
            all_ratings.append([name,score, doc])
```

```
51717it [00:29, 1749.01it/s]
```

In [ ]:
```
import re
rating_df=pd.DataFrame(all_ratings,columns=['name','rating','review'])
rating_df['review']=rating_df['review'].apply(lambda x : re.sub('[^a-zA-Z
```

In [ ]:
```
# len(rating_df)
rating_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1319968 entries, 0 to 1319967
Data columns (total 3 columns):
 #   Column  Non-Null Count    Dtype  
---  ------  --------------    -----  
 0   name    1319968 non-null  object 
 1   rating  1319968 non-null  float64
 2   review  1319968 non-null  object 
dtypes: float64(1), object(2)
memory usage: 30.2+ MB
```

In [ ]:

```python
import pandas as pd
import re
import torch
from torch.utils.data import Dataset, DataLoader
from transformers import BertTokenizer, BertModel, AdamW
import torch.nn as nn
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from tqdm import tqdm
import numpy as np



# Tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

# Dataset class
class RatingDataset(Dataset):
    def __init__(self, reviews, ratings, tokenizer, max_len=512):
        self.reviews = reviews
        self.ratings = ratings - 1
        self.tokenizer = tokenizer
        self.max_len = max_len

    def __len__(self):
        return len(self.reviews)

    def __getitem__(self, item):
        review = str(self.reviews[item])
        rating = self.ratings[item]

        encoding = self.tokenizer.encode_plus(
            review,
            max_length=self.max_len,
            add_special_tokens=True,
            padding='max_length',
            truncation=True,
            return_attention_mask=True,
            return_tensors='pt',
        )

        return {
            'review_text': review,
            'input_ids': encoding['input_ids'].flatten(),
            'attention_mask': encoding['attention_mask'].flatten(),
            'rating': torch.tensor(rating, dtype=torch.long)
        }

# Data Loaders
```

```
def create_data_loader(df, tokenizer, max_len, batch_size):
    ds = RatingDataset(
        reviews=df.review.to_numpy(),
        ratings=df.rating.to_numpy(),
        tokenizer=tokenizer,
        max_len=max_len
    )

    return DataLoader(
        ds,
        batch_size=batch_size,
        num_workers=2
    )

# Split the data
train_df, val_df = train_test_split(rating_df[:10000], test_size=0.1)
train_data_loader = create_data_loader(train_df, tokenizer, 512, 16)
val_data_loader = create_data_loader(val_df, tokenizer, 512, 16)

# BERT Model with a linear layer
class RatingClassifier(nn.Module):
    def __init__(self, n_classes):
        super(RatingClassifier, self).__init__()
        self.bert = BertModel.from_pretrained('bert-base-uncased')
        self.drop = nn.Dropout(p=0.3)
        self.out = nn.Linear(self.bert.config.hidden_size, n_classes)

    def forward(self, input_ids, attention_mask):
        _, pooled_output = self.bert(
            input_ids=input_ids,
            attention_mask=attention_mask,
            return_dict=False
        )
        output = self.drop(pooled_output)
        return self.out(output)

model = RatingClassifier(n_classes=5)
optimizer = AdamW(model.parameters(), lr=2e-5, correct_bias=False)

# Training function
def train_epoch(model, data_loader, optimizer, device, n_examples):
    model = model.train()
    losses = []
    correct_predictions = 0

    for d in tqdm(data_loader):
        input_ids = d["input_ids"].to(device)
        attention_mask = d["attention_mask"].to(device)
        targets = d["rating"].to(device)

        outputs = model(
            input_ids=input_ids,
            attention_mask=attention_mask
        )

        _, preds = torch.max(outputs, dim=1)
        loss = nn.CrossEntropyLoss()(outputs, targets)

        correct_predictions += torch.sum(preds == targets)
        losses.append(loss.item())
```

```
        loss.backward()
        nn.utils.clip_grad_norm_(model.parameters(), max_norm=1.0)
        optimizer.step()
        optimizer.zero_grad()

    return correct_predictions.double() / n_examples, np.mean(losses)

# Evaluation function
def eval_model(model, data_loader, device, n_examples):
    model = model.eval()
    losses = []
    correct_predictions = 0

    with torch.no_grad():
        for d in data_loader:
            input_ids = d["input_ids"].to(device)
            attention_mask = d["attention_mask"].to(device)
            targets = d["rating"].to(device)

            outputs = model(
                input_ids=input_ids,
                attention_mask=attention_mask
            )

            _, preds = torch.max(outputs, dim=1)
            loss = nn.CrossEntropyLoss()(outputs, targets)

            correct_predictions += torch.sum(preds == targets)
            losses.append(loss.item())

    return correct_predictions.double() / n_examples, np.mean(losses)

# Set up CUDA device for training on GPU
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
model = model.to(device)

# Training loop
epochs = 5

for epoch in range(epochs):
    print(f'Epoch {epoch + 1}/{epochs}')
    train_acc, train_loss = train_epoch(
        model,
        train_data_loader,
        optimizer,
        device,
        len(train_df)
    )
    print(f'Train loss {train_loss}, accuracy {train_acc}')

    val_acc, val_loss = eval_model(
        model,
        val_data_loader,
        device,
        len(val_df)
    )
    print(f'Validation loss {val_loss}, accuracy {val_acc}')
```

```
/usr/local/lib/python3.10/dist-packages/transformers/optimization.py:41
1: FutureWarning: This implementation of AdamW is deprecated and will be
removed in a future version. Use the PyTorch implementation torch.optim.
AdamW instead, or set `no_deprecation_warning=True` to disable this warn
ing
  warnings.warn(
Epoch 1/5
100%|██████████| 563/563 [12:58<00:00,  1.38s/it]
Train loss 0.8675117592193305, accuracy 0.6445555555555555

Validation loss 0.5855241363483762, accuracy 0.782
Epoch 2/5
100%|██████████| 563/563 [13:00<00:00,  1.39s/it]
Train loss 0.4521834301972368, accuracy 0.8391111111111111

Validation loss 0.596233961305448, accuracy 0.798
Epoch 3/5
100%|██████████| 563/563 [13:00<00:00,  1.39s/it]
Train loss 0.28025837784854707, accuracy 0.907888888888889

Validation loss 0.6125108234229542, accuracy 0.844
Epoch 4/5
100%|██████████| 563/563 [13:00<00:00,  1.39s/it]
Train loss 0.21496830662037536, accuracy 0.9316666666666668

Validation loss 0.6335938324826578, accuracy 0.843
Epoch 5/5
100%|██████████| 563/563 [13:00<00:00,  1.39s/it]
Train loss 0.16756210611123248, accuracy 0.9481111111111111

Validation loss 0.656573471121697, accuracy 0.854
```

In [ ]:

```
import os
epoch=5
# Saving the model after each epoch
checkpoint_dir = '/content/drive/MyDrive/ECE 225 Project'
torch.save(model.state_dict(), os.path.join(checkpoint_dir, f'model_epoch
print(f'Model saved to {checkpoint_dir}/model_epoch_{epoch+1}.pth')
```

```
Model saved to /content/drive/MyDrive/ECE 225 Project/model_epoch_6.pth
```