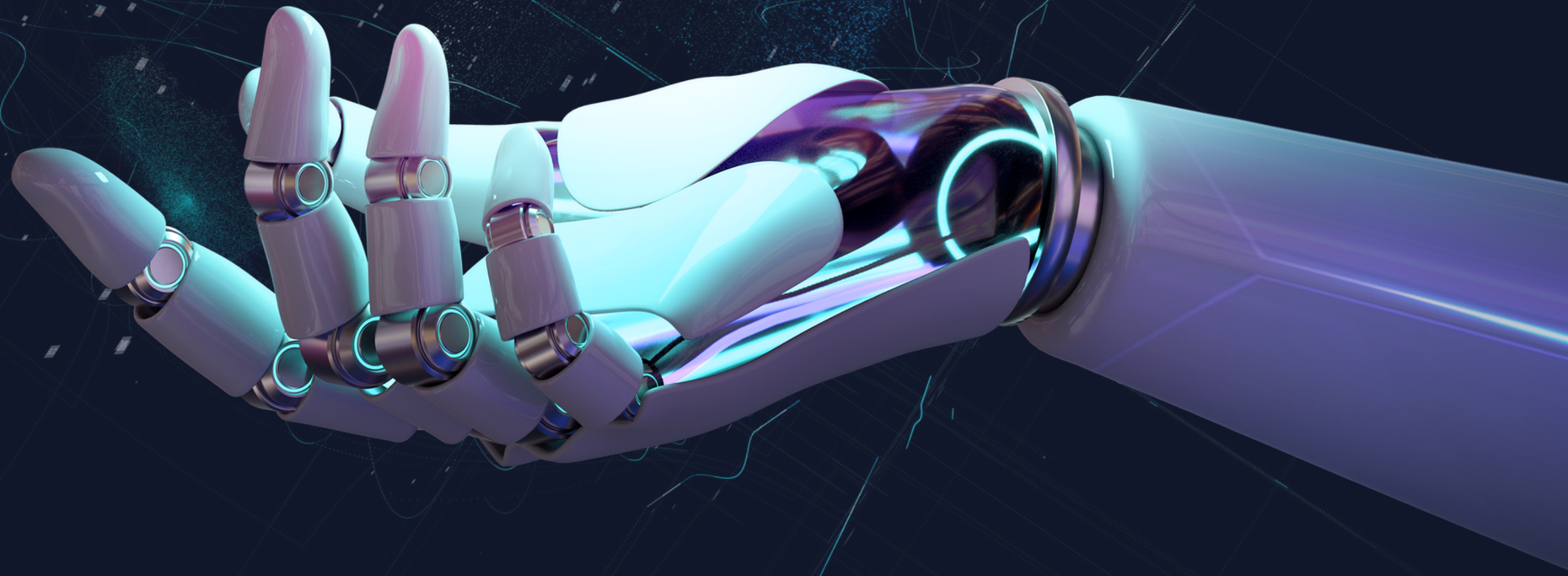
- 尚荣 1820201064
- 陈安赐 1820202031
- 成勇今 1820202040
- 韩瑞 1820202043

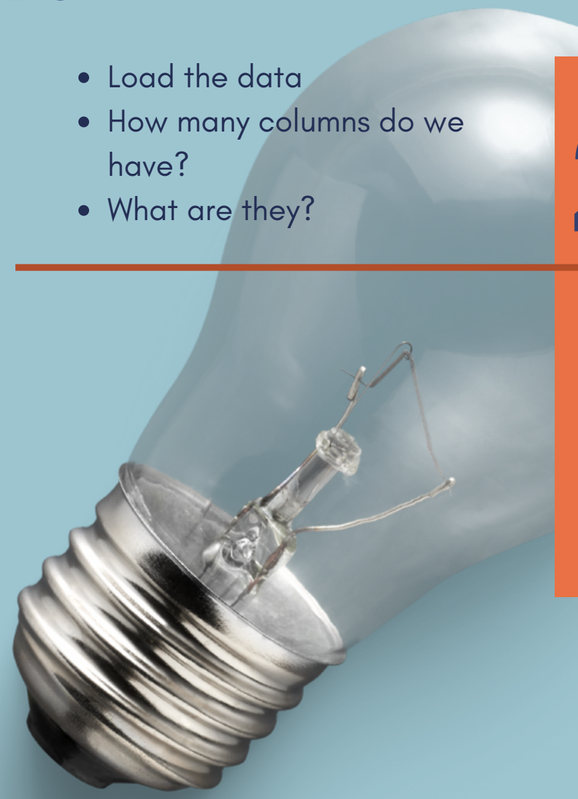
Predict Restaurant's success

GROUP 15



北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

Table of contents



1. Zomato database and overview

- Load the data
- How many columns do we have?
- What are they?

2. Data Preprocessing

- Getting all NAN features
- Missing Values by feature
- Preparing Approx_cost column
- Preparing rate_num column

3. Data Analysis

- How many types of restaurants do we have?
- Top 5 Most/Less Voted Restaurants
- Top 5 Most expensive restaurants
- Book Table Service or Online Service
- Finding Best budget Restaurants in any location
- Geographical analysis

4. Predicting the success of a restaurant

- Defining a custom threshold for splitting restaurants into good and bad
- Feature Extraction

Zomato CSV file and overview

Zomato

Data Analysis Project

kaggle.com

The Zomato CSV file contains information about thousands of restaurants listed on the Zomato platform. The file includes information such as the name and address of the restaurant, its rating, the cuisine type, and the price range.

The Zomato CSV file also can give us information about the geographical location of the restaurants, including latitude and longitude coordinates through API. This information can be used to plot the restaurants on a map, allowing for location-based analysis and visualizations.


```
RangeIndex: 51717 entries, 0 to 51716
Data columns (total 17 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   url                          51717 non-null  object
 1   address                      51717 non-null  object
 2   name                         51717 non-null  object
 3   online_order                 51717 non-null  object
 4   book_table                   51717 non-null  object
 5   rate                         43942 non-null  object
 6   votes                        51717 non-null  int64
 7   phone                        50509 non-null  object
 8   location                     51696 non-null  object
 9   rest_type                    51490 non-null  object
 10  dish_liked                   23639 non-null  object
 11  cuisines                     51672 non-null  object
 12  approx_cost(for two people)  51371 non-null  object
 13  reviews_list                 51717 non-null  object
 14  menu_item                    51717 non-null  object
 15  listed_in(type)              51717 non-null  object
 16  listed_in(city)              51717 non-null  object
dtypes: int64(1), object(16)
```

url: contains the url of the restaurant in the zomato website;
address: contains the address of the restaurant in Bengaluru;
name: contains the name of the restaurant;
online-order: whether online ordering is available in the restaurant or not;
book-table: table book option available or not;
rate: contains the overall rating of the restaurant out of 5;
votes: contains total number of rating for the restaurant as of the above mentioned date;
phone: contains the phone number of the restaurant;
location: contains the neighborhood in which the restaurant is located;
rest-type: restaurant type.

All NAN features

```
feature_na=[feature for feature in df.columns if df[feature].isnull().sum()>0]
feature_na
```

✓ 0.0s Python

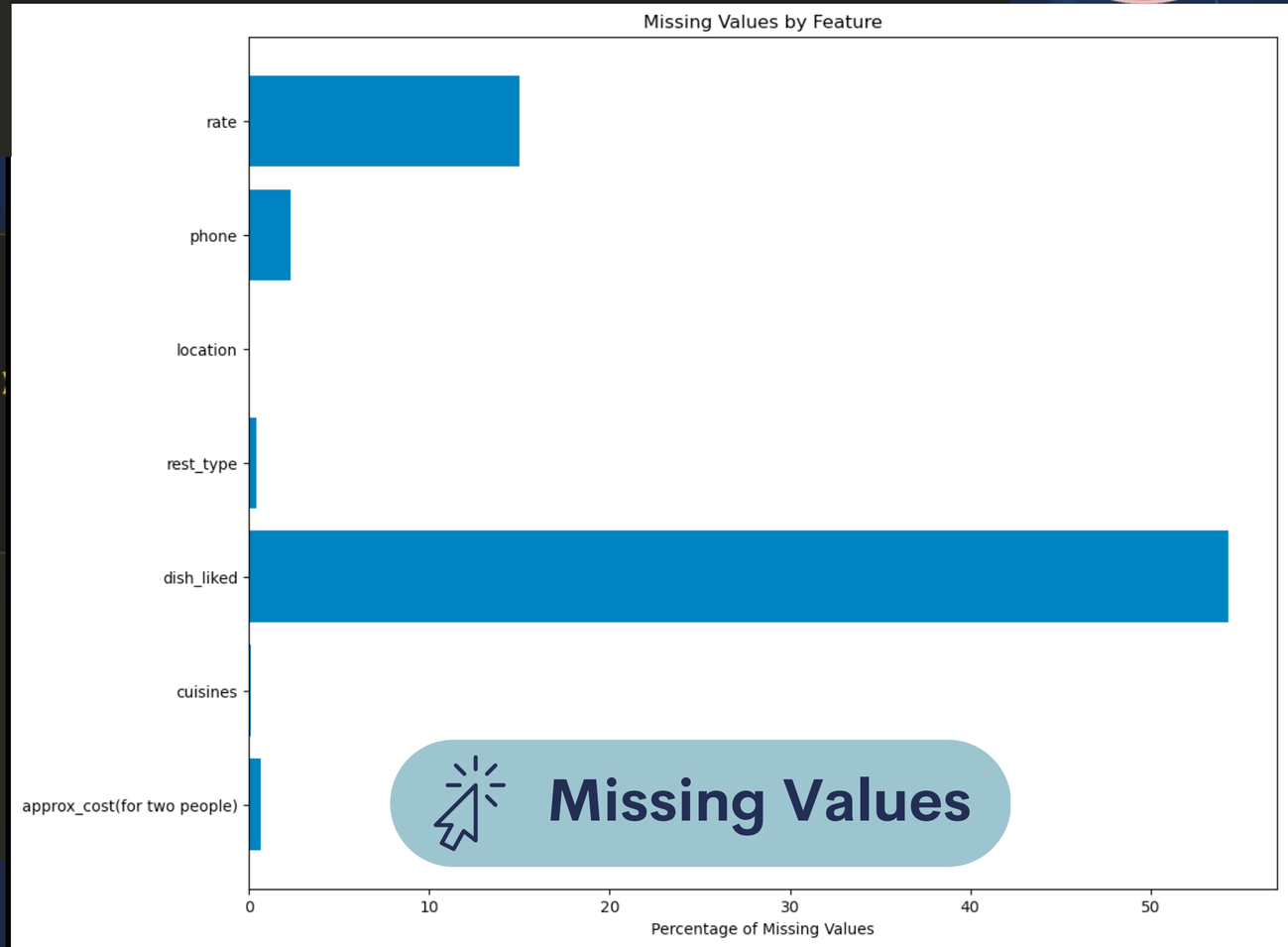
```
['rate',
 'phone',
 'location',
 'rest_type',
 'dish_liked',
 'cuisines',
 'approx_cost(for two people)']
```

```
#% of missing values
import numpy as np
for feature in feature_na:
    print('{} has {} % missing values'.format(feature,np.round(df[feature].isnull()
```

✓ 0.0s Python

```
rate has 15.0337 % missing values
phone has 2.3358 % missing values
location has 0.0406 % missing values
rest_type has 0.4389 % missing values
dish_liked has 54.2916 % missing values
cuisines has 0.087 % missing values
approx_cost(for two people) has 0.669 % missing values
```

```
features = ['rate', 'phone', 'location', 'rest_type',
            'dish_liked', 'cuisines', 'approx_cost(for two people)']
missing_values = [15.0337, 2.3358, 0.0406, 0.4389, 54.2916, 0.087, 0.669]

# Create horizontal bar chart
fig, ax = plt.subplots(figsize=(12, 10))
y_pos = np.arange(len(features))
ax.barh(y_pos, missing_values, align='center')
ax.set_yticks(y_pos)
ax.set_yticklabels(features)
ax.invert_yaxis()  # labels read top-to-bottom
ax.set_xlabel('Percentage of Missing Values')
ax.set_title('Missing Values by Feature')
plt.show()
```



Missing Values

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

```
df['rate'].unique()
```
✓ 0.0s

```
array(['4.1/5', '3.8/5', '3.7/5', '3.6/5', '4.6/5', '4.0/5', '4.2/5',
       '3.9/5', '3.1/5', '3.0/5', '3.2/5', '3.3/5', '2.8/5', '4.4/5',
       '4.3/5', 'NEW', '2.9/5', '3.5/5', nan, '2.6/5', '3.8 /5', '3.4/5',
       '4.5/5', '2.5/5', '2.7/5', '4.7/5', '2.4/5', '2.2/5', '2.3/5',
       '3.4 /5', '-', '3.6 /5', '4.8/5', '3.9 /5', '4.2 /5', '4.0 /5',
       '4.1 /5', '3.7 /5', '3.1 /5', '2.9 /5', '3.3 /5', '2.8 /5',
       '3.5 /5', '2.7 /5', '2.5 /5', '3.2 /5', '2.6 /5', '4.5 /5',
       '4.3 /5', '4.4 /5', '4.9/5', '2.1/5', '2.0/5', '1.8/5', '4.6 /5',
       '4.9 /5', '3.0 /5', '4.8 /5', '2.3 /5', '4.7 /5', '2.4 /5',
       '2.1 /5', '2.2 /5', '2.0 /5', '1.8 /5'], dtype=object)
```

Preparing rate_num column

Get rid of the slashes in data and take the only first part of it. At the end, conert the entire data column into float so that we can perform data analysis or algorithms in the future.

```
df['rate'][0].split('/')[0]
```
✓ 0.0s

```
'4.1'
```

```
def split(x):
    return x.split('/')[0]
```
✓ 0.0s

```
df['rate'].dtype
```
✓ 0.0s

```
dtype('O')
```

```
df['rate'].isnull().sum()
```
✓ 0.0s

```
7775
```

```
### right now it has some NAN Values so it will be of float data-type,
### we have to split it & access
df['rate']=df['rate'].astype(str).apply(split)
### ''' df['rate'] = df['rate'].astype(str).apply(lambda x: x.split('
```
✓ 0.0s

```
df['rate'].replace('NEW',0,inplace=True)
df['rate'].replace('-',0,inplace=True)
```
✓ 0.0s

```
df['rate']=df['rate'].astype(str).astype(float)
```
✓ 0.0s

Preparing Approx_cost column

This line converts the 'approx_cost(for two people)' column to a string data type using the .astype(str) method. Then, it applies a lambda function using the .apply() method to each value in the column to replace any commas with an empty string using the .replace() method. Finally, it saves the modified column back to the DataFrame with the same column name.

```
### right now it has some NAN Values so it will be of float data-type,dats why very first we have to convert it into str
### we have to remove this comma
df['approx_cost(for two people)'] = df['approx_cost(for two people)'].astype(str).apply(lambda x: x.replace(',', ''))
```
✓ 0.0s

```
df['approx_cost(for two people)']=df['approx_cost(for two people)'].astype(float)
```
✓ 0.0s

```
df['approx_cost(for two people)'].dtype
```
✓ 0.0s

```
dtype('float64')
```

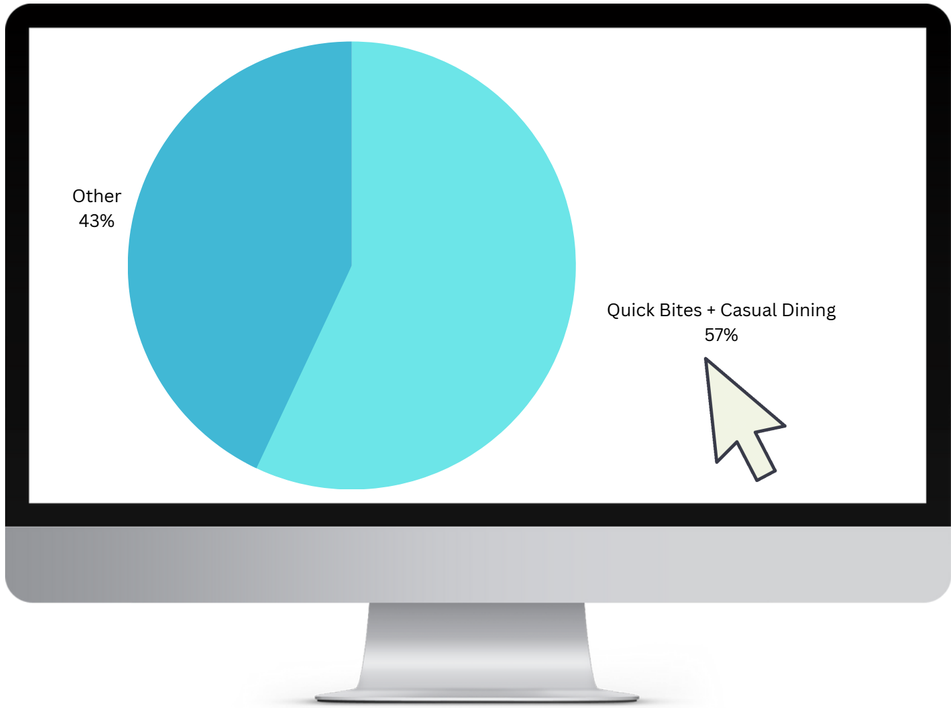
```
import matplotlib.pyplot as plt
plt.figure(figsize=(20,12))
df['rest_type'].value_counts().nlargest(20).plot.bar(color='red')

### to provide styling to text on x-axis
plt.gcf().autofmt_xdate()
```
✓ 0.2s

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

How many types of restaurants we have?

```
def mark(x):
    if x in ('Quick Bites', 'Casual Dining'):
        return 'Quick Bites + Casual Dining'
    else:
        return 'other'
```
✓ 0.0s

```
df['Top_types']=df['rest_type'].apply(mark)
```
✓ 0.0s



Aggregate functions

grouping the dataframe 'df' by the 'name' column and applying several aggregation functions on the remaining columns. Specifically, summing up the 'votes' column, counting the number of unique 'url' values, calculating the mean of 'approx_cost(for two people)' column, and calculating the mean of the 'rate' column.

```
rest=df.groupby('name').agg({'votes': 'sum','url': 'count','approx_cost(for two people)': 'mean','rate': 'mean'}).reset_index()
rest
```

✓ 0.0s

	name	votes	url	approx_cost(for two people)	rate
0	#FeelTheROLL	14	2	200.0	3.400000
1	#L-81 Cafe	432	9	400.0	3.900000
2	#Vibes Restro	0	3	700.0	NaN
3	#refuel	111	3	400.0	3.700000
4	'Brahmins' Thatte Idli	0	1	100.0	NaN
...	...	...	...	...	...
8787	late100	0	5	200.0	NaN
8788	nu.tree	1443	8	400.0	4.314286
8789	re:cess - Hilton Bangalore Embassy GolfLinks	438	3	1200.0	4.100000
8790	repEAT Hub	0	2	200.0	NaN
8791	sCoolMeal	0	5	300.0	NaN

8792 rows × 5 columns

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

```
rest.columns = ['name', 'total_votes', 'total_unities', 'avg_approx_cost',
rest.head()
```

✓ 0.0s

	name	total_votes	total_unities	avg_approx_cost	mean_rating
0	#FeelTheROLL	14	2	200.0	3.4
1	#L-81 Cafe	432	9	400.0	3.9
2	#Vibes Restro	0	3	700.0	NaN
3	#refuel	111	3	400.0	3.7
4	'Brahmins' Thatte Idli	0	1	100.0	NaN

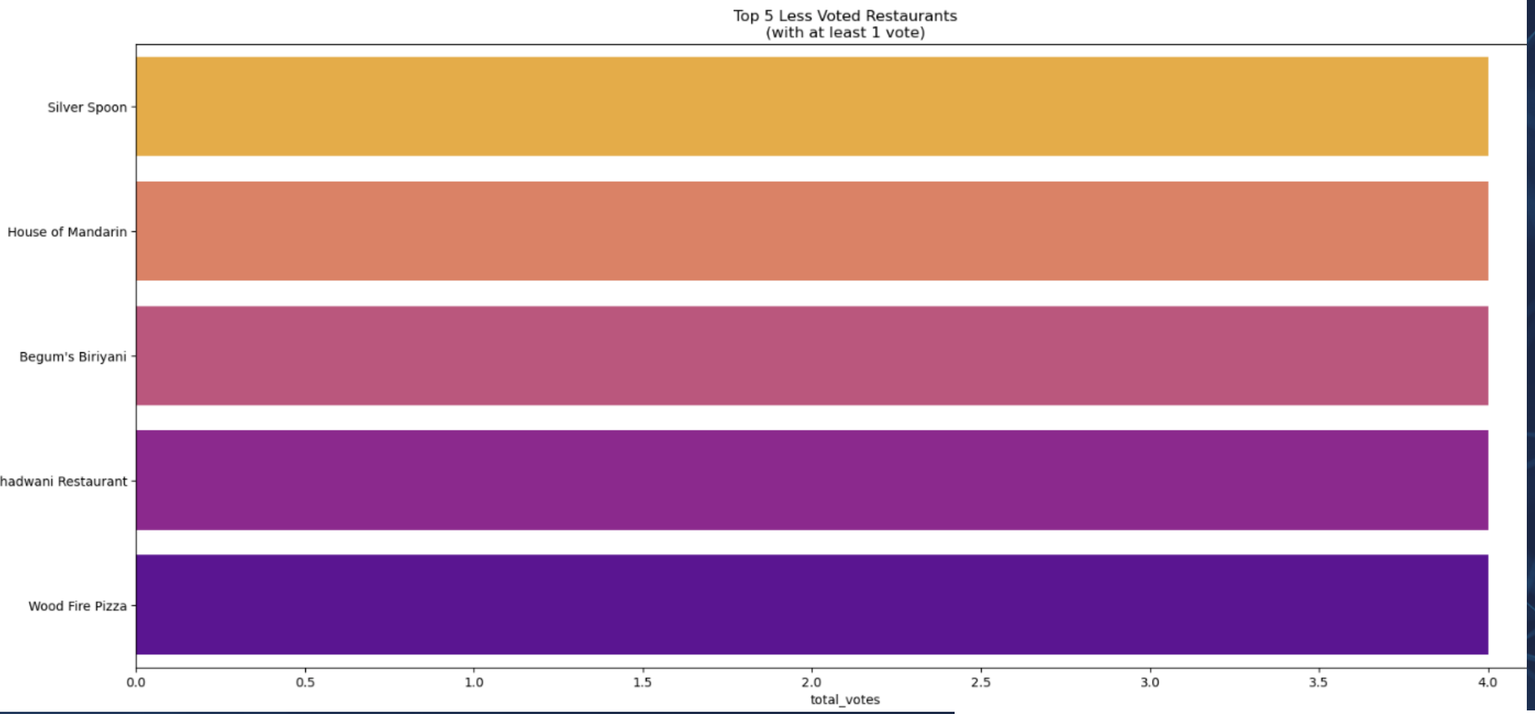
```
rest['votes_per_unity'] = rest['total_votes'] / rest['total_unities']
rest.head()
```

✓ 0.0s

	name	total_votes	total_unities	avg_approx_cost	mean_rating	votes_per_unity
0	#FeelTheROLL	14	2	200.0	3.4	7.0
1	#L-81 Cafe	432	9	400.0	3.9	48.0
2	#Vibes Restro	0	3	700.0	NaN	0.0
3	#refuel	111	3	400.0	3.7	37.0
4	'Brahmins' Thatte Idli	0	1	100.0	NaN	0.0

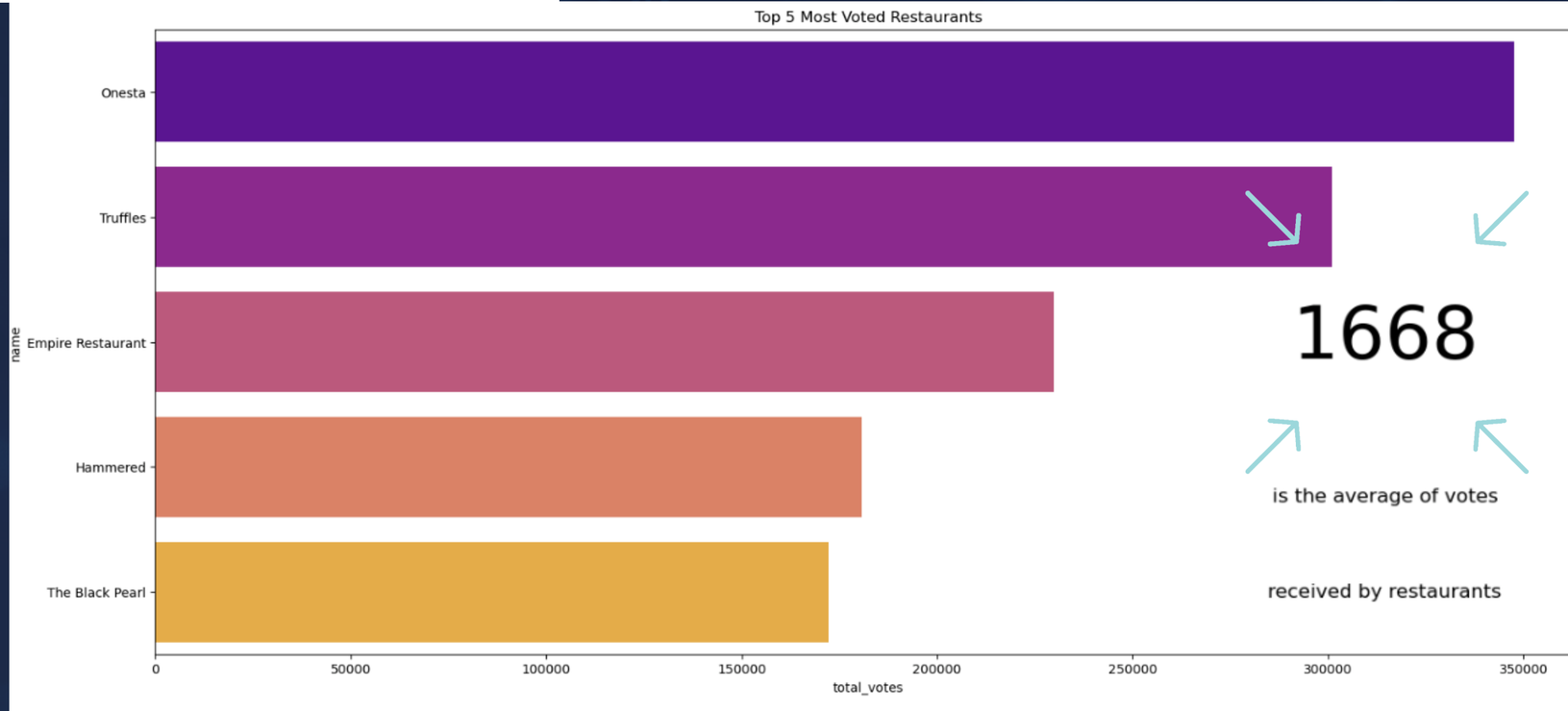
```
import seaborn as sns
# Creating a figure for restaurants overview analysis
fig, (ax1,ax2,ax3) = plt.subplots(3,1, figsize=(20,30))

# Plot Pack 01 - Most popular restaurants (votes)

# Annotations
ax1.text(0.50, 0.30, int(popular['total_votes'].mean()), fontsize=45, ha='center')
ax1.text(0.50, 0.12, 'is the average of votes', fontsize=12, ha='center')
ax1.text(0.50, 0.00, 'received by restaurants', fontsize=12, ha='center')
ax1.axis('off')

sns.barplot(x='total_votes', y='name', data=popular.sort_values(by='total_votes', ascending=False)[0:5],ax=ax2, pale
ax2.set_title('Top 5 Most Voted Restaurants', size=12)

sns.barplot(x='total_votes', y='name', data=popular.sort_values(by='total_votes', ascending=False).query('total_vote
ax3.set_title('Top 5 Less Voted Restaurants\n(with at least 1 vote)', size=12)
```

✓ 0.2s

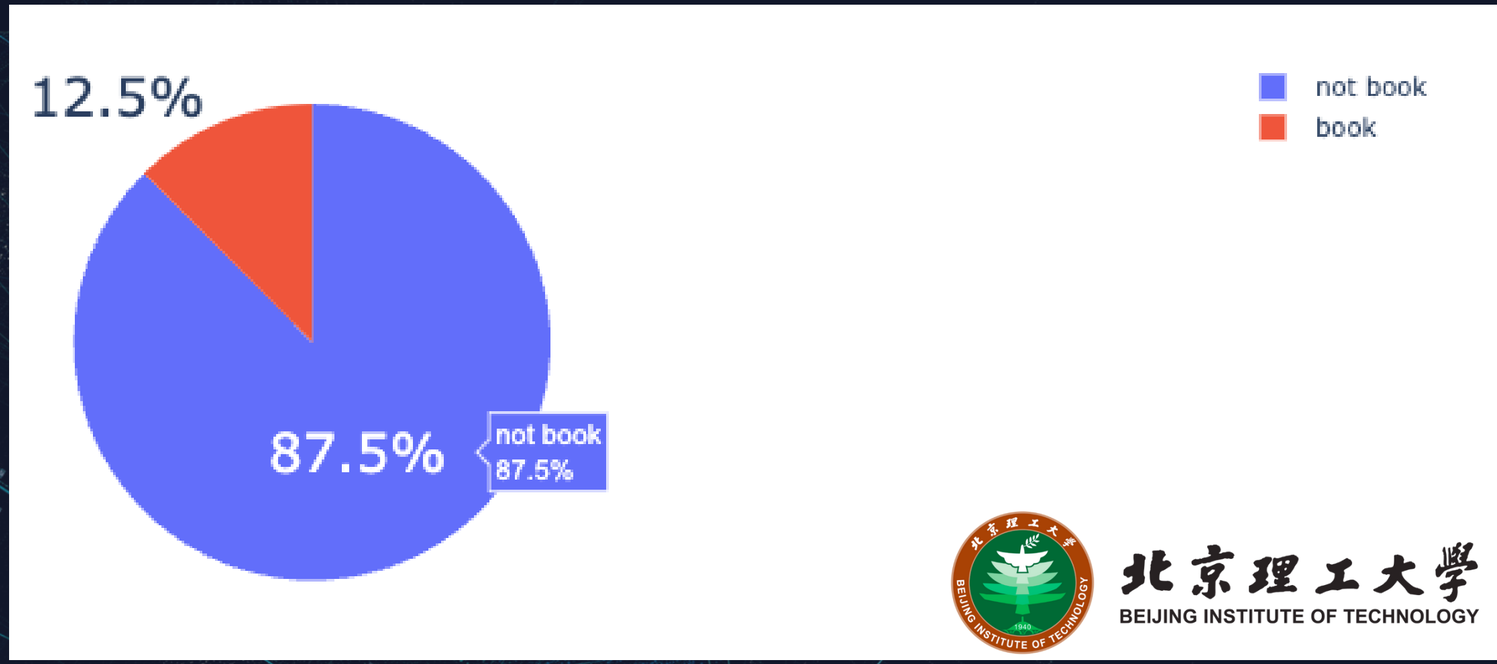

Top 5 Less Voted Restaurants
(with at least 1 vote)
Silver Spoon
House of Mandarin
Begum's Biriyani
hadwani Restaurant
Wood Fire Pizza
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
total_votes


Top 5 Less Voted Restaurants

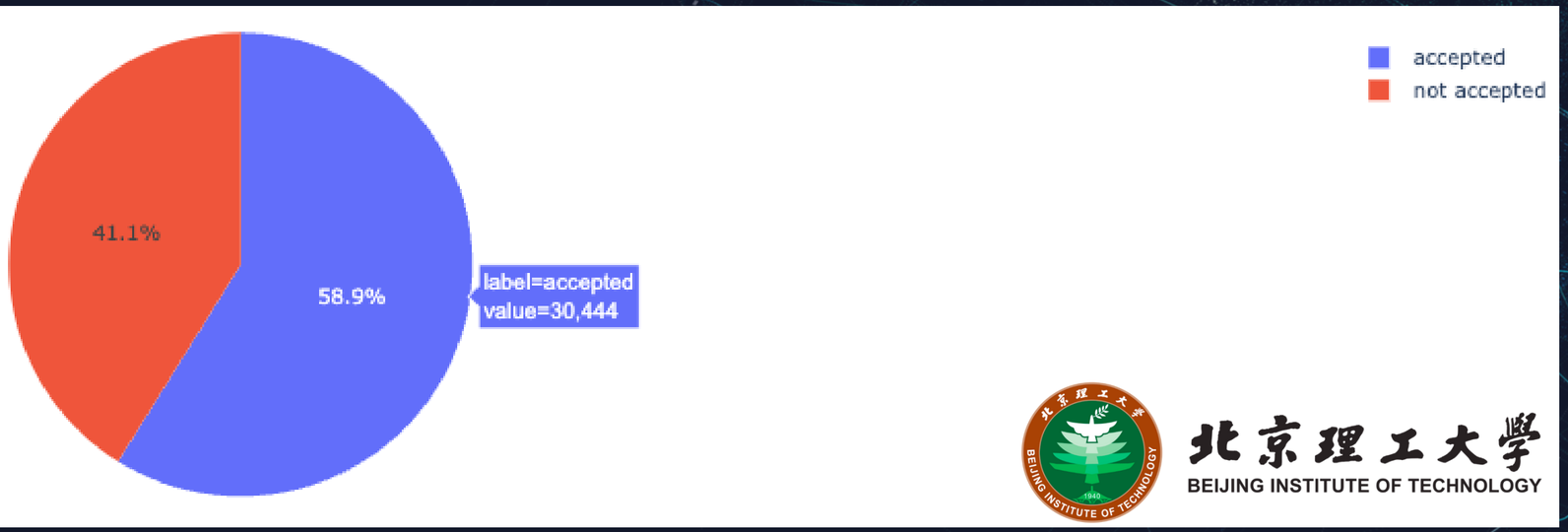

Top 5 Most Voted Restaurants


Top 5 Most Voted Restaurants
Onesta
Truffles
Empire Restaurant
Hammered
The Black Pearl
name
0
50000
100000
150000
200000
250000
300000
350000
total_votes
1668
is the average of votes
received by restaurants

HOW MANY RESTAURANTS OFFER BOOK TABLE SERVICE?



北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY



北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

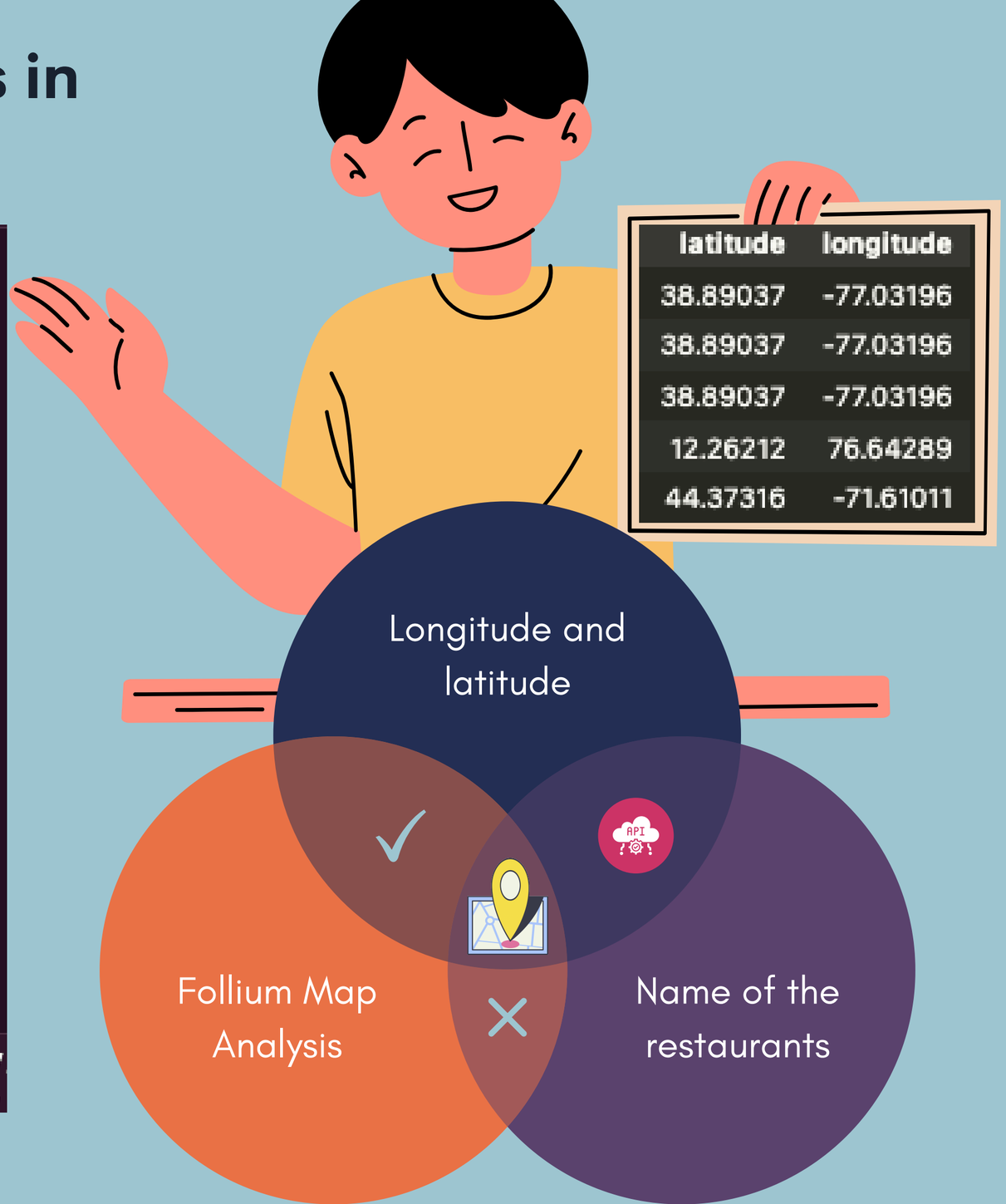
HOW ABOUT ONLINE ORDER SERVICE?

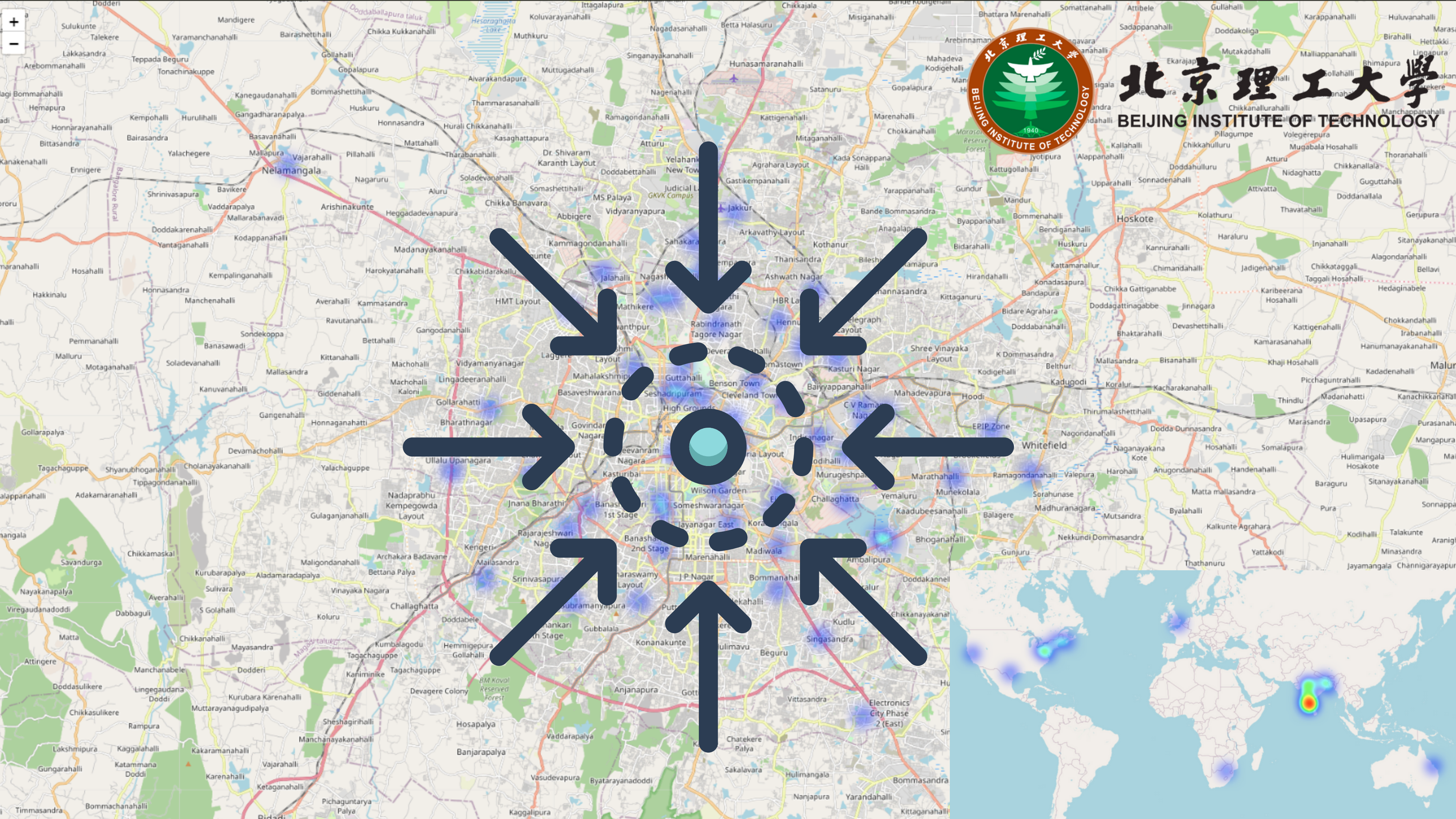
Finding Best budget Restaurants in any location

```python
import requests
# MAPQUEST API
API_KEY = 'FvWB34iwtjSNxakt3jRVLD2bQbrxQsUU'  # Replace with your API key
BASE_URL = 'http://www.mapquestapi.com/geocoding/v1/address'

latitudes = []
longitudes = []

for location in locations['Name']:
    response = requests.get(
        BASE_URL, params={'key': API_KEY, 'location': location})
    if response.status_code == 200:
        data = response.json()
        lat = data['results'][0]['locations'][0]['latLng']['lat']
        lng = data['results'][0]['locations'][0]['latLng']['lng']
        latitudes.append(lat)
        longitudes.append(lng)
    else:
        latitudes.append(None)
        longitudes.append(None)

print(latitudes)
print(longitudes)
```

✓ 1m 44.1s

```
[12.92232, 12.93899, 12.95771, 22.17625, 12.89819, 12.93178, 22.75641, 12.89757
[77.56986, 77.57141, 77.47156, 88.42072, 77.55927, 77.52668, 75.89247, 77.5283,
```




北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY
1940

Predicting the success of a restaurant

How do we make a good prediction?

1. ***Threshold***
2. ***Feature Extraction***
3. ***Encoding***

We have to extract the features that help us to make good predictions. Because all the data can create bias/noise in the data and our prediction may not be accurate.

We can filter our data into categories/classes setting a threshold value. If we set the threshold to 3.4 for rating, then restaurants whose rating is higher than 3.4 are considered good whereas the other ones are bad.

In order to put our data into algorithm training, we have to encode the features because most of the features contain string which is not suitable for machine learning algorithms.

Threshold

```
df['rate'].unique()
```

✓ 0.0s

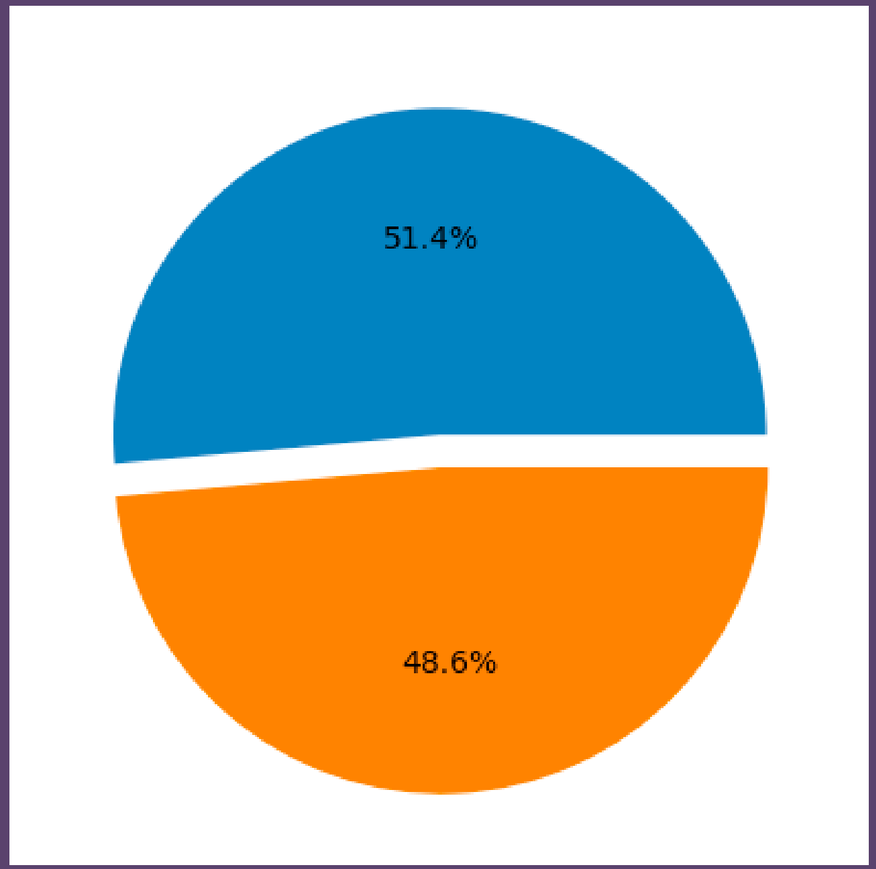
```
array([4.1, 3.8, 3.7, 3.6, 4.6, 4. , 4.2, 3.9, 3.1, 3. , 3.2, 3.3, 2.8,
       4.4, 4.3, 0. , 2.9, 3.5, nan, 2.6, 3.4, 4.5, 2.5, 2.7, 4.7, 2.4,
       2.2, 2.3, 4.8, 4.9, 2.1, 2. , 1.8])
```

```
def assign(x):
    if x>0:
        return 1
    else:
        return 0
df['rated']=df['rate'].apply(assign)
```

✓ 0.0s

```
new_restaurants = df[df['rated'] == 0]
train_val_restaurants = df.query('rated == 1')
```

✓ 0.0s




```
# Defining a custom threshold for splitting restaurants into good and bad
threshold = 3.75
train_val_restaurants['target'] = train_val_restaurants['rate'].apply(lambda x: 1 if x >= threshold else 0)
```

✓ 0.0s

Feature Extraction

```
## train_val_restaurants['total_cuisines'] = train_val_restaurants['cuisines'].astype(str).apply(lambda x: len(x.split(',')))

def count(x):
    return len(x.split(','))
```

✓ 0.0s

```
#### as it have some NAN value that's why, first I have to convert into str  &  then apply a function
train_val_restaurants['total_cuisines']=train_val_restaurants['cuisines'].astype(str).apply(count)
train_val_restaurants['multiple_types']=train_val_restaurants['rest_type'].astype(str).apply(count)
```

✓ 0.0s

```
train_val_restaurants.columns
```

✓ 0.0s

```
Index(['url', 'address', 'name', 'online_order', 'book_table', 'rate', 'votes',
       'phone', 'location', 'rest_type', 'dish_liked', 'cuisines',
       'approx_cost(for two people)', 'reviews_list', 'menu_item',
       'listed_in(type)', 'listed_in(city)', 'Top_types', 'rated', 'target',
       'total_cuisines', 'multiple_types'],
      dtype='object')
```

```
data.isnull().sum()
```

✓ 0.0s

```
online_order                    0
book_table                      0
location                        0
rest_type                     149
multiple_types                  0
total_cuisines                  0
listed_in(type)                 0
listed_in(city)                 0
approx_cost(for two people)   247
target                          0
dtype: int64
```

```
data.dropna(how='any',inplace=True)
```

✓ 0.0s

```
data.isnull().sum()
```

✓ 0.0s

```
online_order                  0
book_table                    0
location                      0
rest_type                     0
multiple_types                0
total_cuisines                0
listed_in(type)               0
listed_in(city)               0
approx_cost(for two people)   0
target                        0
dtype: int64
```

After defining the target and splitting the data into train+val and test sets, let's define the features to be used on training. Here we will take a look at the raw data to select valuable features and apply some steps to create other ones.

```
data = train_val_restaurants[imp_features]
```

✓ 0.0s

```
imp_features=['online_order','book_table','location','rest_type','multiple_types','total_cuisines','listed_in(type)', 'listed_in(city)','approx_cost(for two people)','target']
```

✓ 0.0s

```
# Splitting features by data type
cat_features= [col for col in data.columns if data[col].dtype == 'O']
num_features= [col for col in data.columns if data[col].dtype != 'O']
```
✓ 0.0s

北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

```
cat_features
```
✓ 0.0s

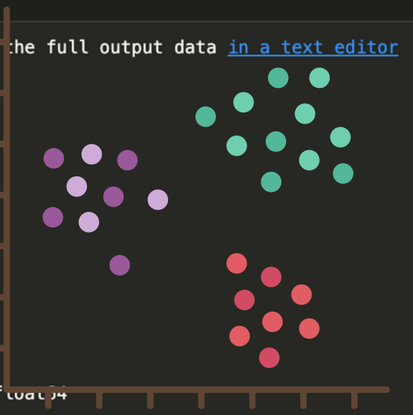
```
['online_order',
 'book_table',
 'location',
 'rest_type',
 'listed_in(type)',
 'listed_in(city)']
```

```
for feature in cat_features:
    print('{} has total {} unique features'.format(feature, data[feature].nunique()))
```
✓ 0.0s

```
online_order has total 2 unique features
book_table has total 2 unique features
location has total 92 unique features
rest_type has total 87 unique features
listed_in(type) has total 7 unique features
listed_in(city) has total 30 unique features
```

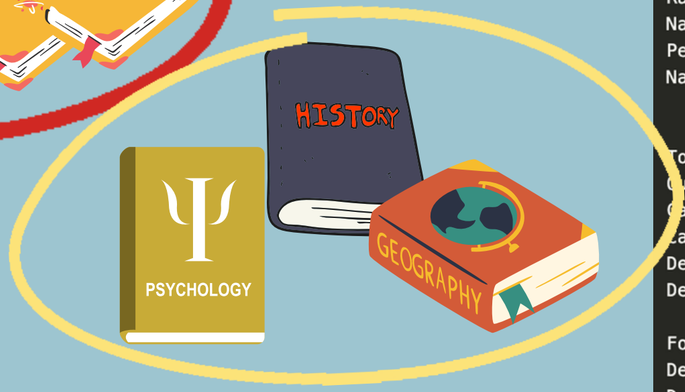
```
cols=['location','rest_type','listed_in(city)']
for col in cols:
    print('Total feature in {} are {}'.format(col,data[col].nunique()))
    print(data[col].value_counts()/(len(data))*100)
    print('\n')
```
✓ 0.0s

```
Output exceeds the size limit. Open the full output data in a text editor
Total feature in location are 92
BTM                      9.398851
Koramangala 5th Block    5.565651
HSR                      4.829057
Indiranagar              4.361416
JP Nagar                 4.143345
                           ...
Yelahanka                0.009692
West Bangalore           0.007269
Rajarajeshwari Nagar     0.004846
Nagarbhavi               0.002423
Peenya                   0.002423
Name: location, Length: 92, dtype: float64


Total feature in rest_type are 87
Quick Bites                   33.643478
Casual Dining                 23.299654
Cafe                           8.163117
Dessert Parlor                 4.482566
Delivery                       4.048848
                                ...
Food Court, Beverage Shop      0.004846
Dessert Parlor, Food Court     0.004846
Dessert Parlor, Kiosk          0.004846
...
New BEL Road             1.378692
Name: listed_in(city), dtype: float64
```



Set a threshold for

- 'location'
- 'rest_type'
- 'listed_in(city)'



FEATURE	THRESHOLD
Location	0.4
Gold Foil Pillow Case	1.5

```python
for feature in cat_features:
    print('{} has total {} unique features'.format(feature, data[feature].nunique()))
```

✓ 0.0s

```
online_order has total 2 unique features
book_table has total 2 unique features
location has total 47 unique features
rest_type has total 11 unique features
listed_in(type) has total 7 unique features
listed_in(city) has total 30 unique features
```

After applying feature reduction, we will observe less number of features

```python
import pandas as pd
data_cat = data[cat_features]
for col in cat_features:
    col_encoded = pd.get_dummies(data_cat[col],prefix=col,drop_first=True) # drop_first=True because it removes the additional column encoding
    data_cat=pd.concat([data_cat,col_encoded],axis=1)
    data_cat.drop(col, axis=1, inplace=True)
```

✓ 0.0s

```python
data_cat.head(10)
```

✓ 0.0s Python

	online_order_Yes	book_table_Yes	location_Banashankari	location_Banaswadi	location_Bannerghatta Road	location_Basavanagudi	location_Bellandur	location_Brigade Road	locatio
0	1	1	1	0	0	0	0	0	
1	1	0	1	0	0	0	0	0	
2	1	0	1	0	0	0	0	0	
3	0	0	1	0	0	0	0	0	

Applying Machine Learning Models

```
# Splitting the data
X = data_final.drop('target', axis=1)
y = data_final['target'].values
```

✓ 0.0s

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.20, random_state=42)
```

✓ 0.3s

```python
# Make predictions on validation dataset

for name, model in models:
    print(name)
    model.fit(X_train, y_train)

    # Make predictions.
    predictions = model.predict(X_test)

    # Compute the error.
    from sklearn.metrics import confusion_matrix
    print(confusion_matrix(predictions, y_test))

    from sklearn.metrics import accuracy_score
    print(accuracy_score(predictions,y_test))
    print('\n')
```

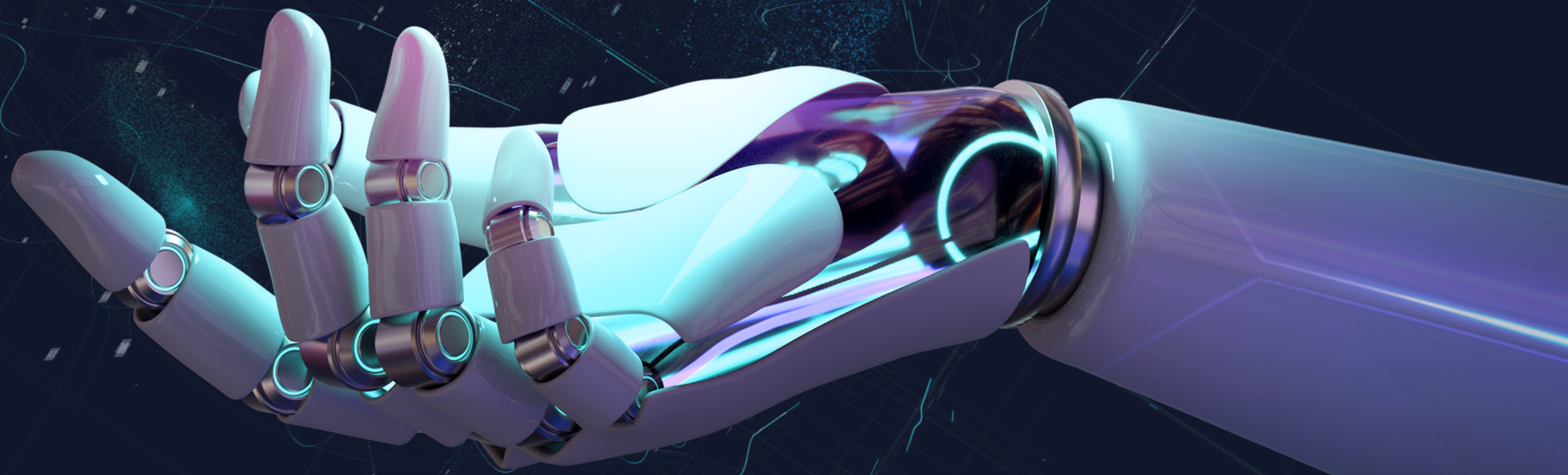
✓ 6.7s


```
LogisticRegression
[[3452 1504]
 [ 800 2499]]
0.7208964264082375


Naive Bayes
[[3040 1460]
 [1212 2543]]
0.6763173834039976


RandomForest
[[3497  964]
 [ 755 3039]]
0.7917625681405209


Decision Tree
[[3649  812]
 [ 603 3191]]
0.8285887341005451


KNN
[[3623 1006]
 [ 629 2997]]
0.8019382192610539
```

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

Q and A

ASK AWAY!



Thank you!