Answer 1-a :- Description of the dataset

Bangalore is most multi-cultural city of India and it is the largest city of Karnataka state. Therefore, it has the restaurants having cuisines from all over the world. It is the best place for food lovers. There are approximately 12000 restaurants in Bangalore. Therefore, it becomes difficult for someone to decide which restaurant is better and which restaurant provides better rates. Also, if anyone wants to open a new restaurant in any location, then they have tough competition deciding which cuisine to choose and what should be the range of approximate cost. Therefore, this Zomato dataset aim to analyse restaurants according to their demography. Also, we can predict the approximate cost for 2 people based on certain number of attributes.

Zomato dataset has following 17 attributes:
url: It shows the url of the restaurant. Each restaurant is assigned a unique url on the Zomato website. There are 51717 unique values of url in the dataset.
address: It contains the address of the restaurant. There are 11495 unique values in address column.
name: It shows the name of the restaurant. There are 8792 unique values for this column.
online_order: It shows whether the restaurant accepts online order or not. There are approximately 30000 restaurants which accepts online ordering and approximate 22000 restaurants which doesn’t accept the online order.
book_table: It shows whether table booking is available at the restaurant. There are approximately 45000 restaurants which allows for table booking and approximate 7000 restaurants which doesn’t allow table booking.
rate: It shows the rating of the restaurant out of 5 stars. It has some values as “NEW”, “-” and null, which means that the restaurant is not rated yet.
votes: It shows how many numbers of votes the restaurant has received. Mostly all the restaurant has number of votes between 0 and 1000.
phone: It shows the phone number of the restaurant. Only 2% values are null in this column.
location: It shows the neighbourhood in which the restaurant is located. 10% of restaurant are in the BTM neighbourhood and HSR neighbourhood has 5 % of the total restaurants.
rest_type: It shows the type of the restaurant. This column contains values which are comma separated. For example, value “Beverage Shop, Quick Bites” means that the restaurant server Quick Bites and Beverages both.
dish_liked: This column shows the dishes liked by people in this restaurant. 54% values are null in this column.
cuisines: It shows the cuisines server by the restaurant. This column also contains comma separated values. For example, value “Chinese, North Indian, Thai” means that the restaurant server Chinese, North Indian, and Thai.
approx_cost: It shows the approximate cost for two people at the restaurant.
reviews_list: It shows the review posted by customer for the restaurant. It contains rating and the review comment.
menu_item: It shows the menus available at the restaurant.
listed_in(type): It shows the type of meal. 50% values are “Delivery” and 34% values are “Dine-out”. Rest all the values are combined in 16%.
listed_in(city): It shows the neighbourhood in which the restaurant is located.

In [1]:

```
# Import the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# %matplotlib inline
```

Answer 2-a :- Loading the dataset

Dataset has been loaded using pandas "read_csv" function as shown in below "Code Cell". The we are displaying the first 10 rows of the dataset using "function" fucntion. The data loaded is stored in a pandas dataframe "df"

In [3]:

```
# #  loading the dataset
df=pd.read_csv("zomato.csv")
print("Total Attributes",df.shape)
df.head(10)
# df = pd.read_csv('/content/zomato-bangalore-restaurants.zip', compression='zip', head
er=0, sep=',', quotechar='"')
# print("Attributes",df.columns)
# print("Total Attributes",df.shape)
# df.head(10)
```

```
Total Attributes (51717, 17)
```

Out[3]:

	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	
5	https://www.zomato.com/bangalore/timepass-dinn...	37, 5-1, 4th Floor, Bosco Court, Gandhi Bazaar...	Timepass Dinner	Yes	
6	https://www.zomato.com/bangalore/rosewood-inte...	19/1, New Timberyard Layout, Beside Satellite ...	Rosewood International Hotel - Bar & Restaurant	No	
7	https://www.zomato.com/bangalore/onesta-banash...	2469, 3rd Floor, 24th Cross, Opposite BDA Comp...	Onesta	Yes	
8	https://www.zomato.com/bangalore/penthouse-caf...	1, 30th Main Road, 3rd Stage, Banashankari, Ba...	Penthouse Cafe	Yes	
9	https://www.zomato.com/bangalore/smacznego-ban...	2470, 21 Main Road, 25th Cross, Banashankari, ...	Smacznego	Yes	

Showing the information of each attribute

The "info" function shows details of each attribute. It helps in identifying number of null and non values.

In [4]:

```
# Showing the information of dataframe
df.info()
```

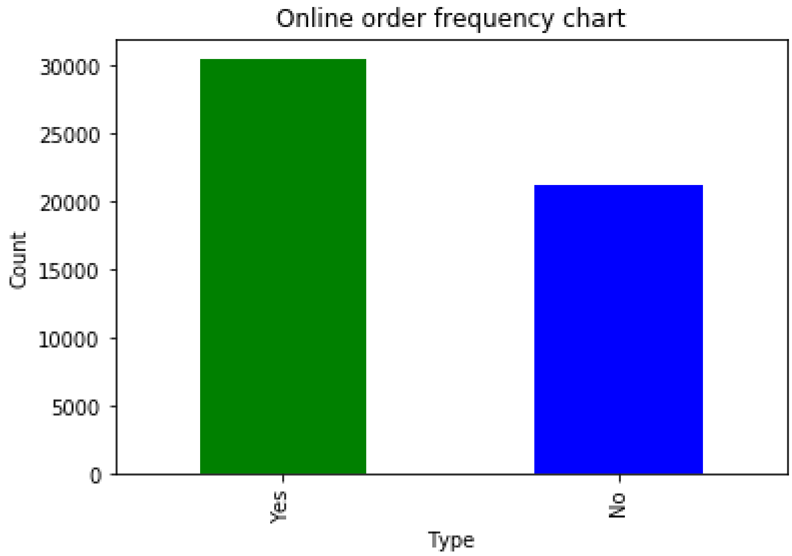
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51717 entries, 0 to 51716
Data columns (total 17 columns):
url                            51717 non-null object
address                        51717 non-null object
name                           51717 non-null object
online_order                   51717 non-null object
book_table                     51717 non-null object
rate                           43942 non-null object
votes                          51717 non-null int64
phone                          50509 non-null object
location                       51696 non-null object
rest_type                      51490 non-null object
dish_liked                     23639 non-null object
cuisines                       51672 non-null object
approx_cost(for two people)    51371 non-null object
reviews_list                   51717 non-null object
menu_item                      51717 non-null object
listed_in(type)                51717 non-null object
listed_in(city)                51717 non-null object
dtypes: int64(1), object(16)
memory usage: 6.7+ MB
```

Answer 2-b :- Frequency distribution of different attributes.

Following are the attributes for which frequency plots are shown in below graphs:
online_order: This shows whether the restaurant accepts online order or not. Bar graph has been plotted for this attribute as shown below. From the graph we can say that, there are approximately 30000 restaurants which accepts online ordering and approximate 22000 restaurants which doesn’t accept the online order.
book_table: This attribute shows whether table booking facility is available or not for the restaurant. Bar graph has been plotted for this attribute as shown below. From the graph we can say that, there are approximately 45000 restaurants which allows for table booking and approximate 7000 restaurants which doesn’t allow table booking.
votes: This attribute shows the number of votes the restaurant has received. This attribute is not a categorical value. So, histogram has been plotted for “votes” as shown below. From the graph we can say that, mostly all the restaurant has number of votes between 0 and 1000.
location: This attribute shows the location of the restaurant. Bar graph has been plotted for this attribute as shown below. From the graph we can say that, there are approximately 5200 restaurants in BTM location.

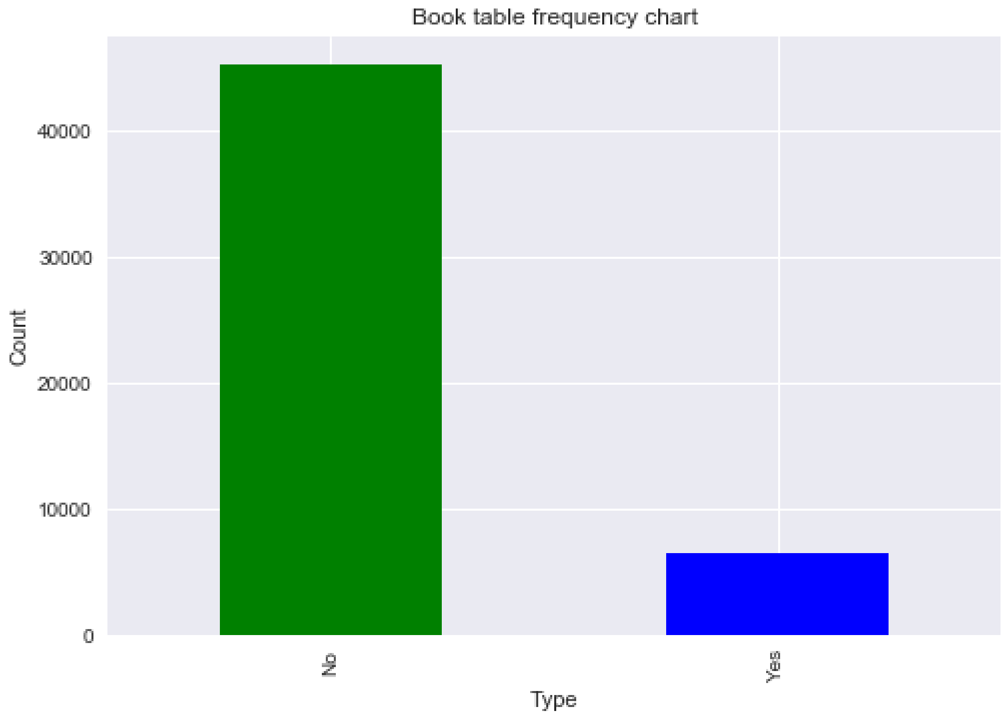
In [5]:

```
df['online_order'].value_counts().plot(kind='bar',title="Online order frequency chart",
color=tuple(["g", "b","r","y","k"]))
plt.xlabel('Type')
plt.ylabel('Count')
plt.show()
```



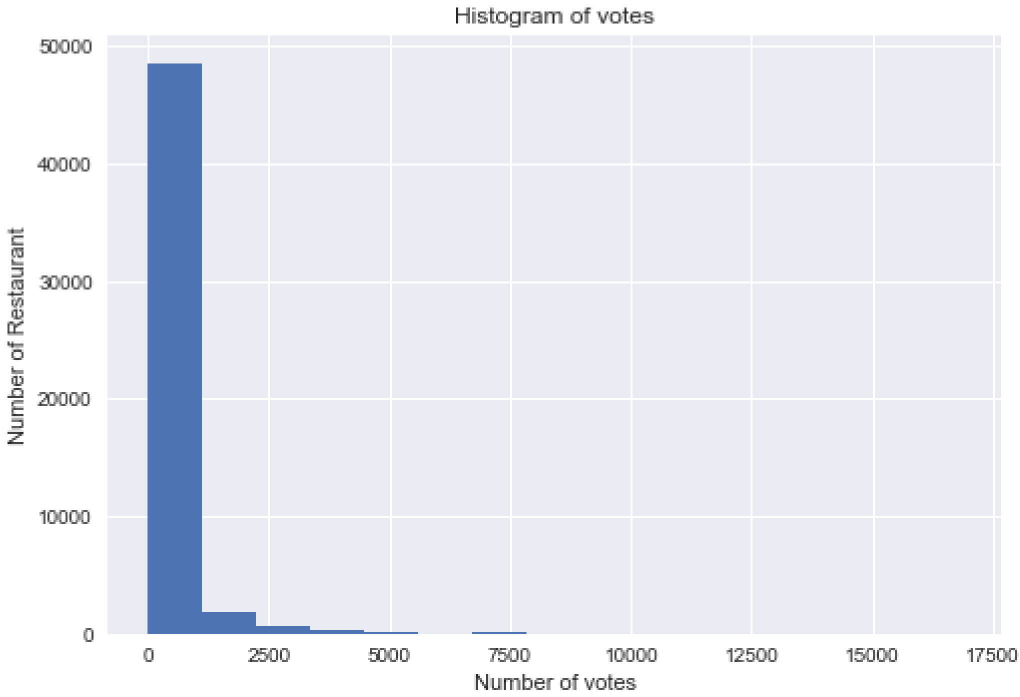
In [0]:

```
df['book_table'].value_counts().plot(kind='bar',title="Book table frequency chart",colo
r=tuple(["g", "b","r","y","k"]))
plt.xlabel('Type')
plt.ylabel('Count')
plt.show()
```



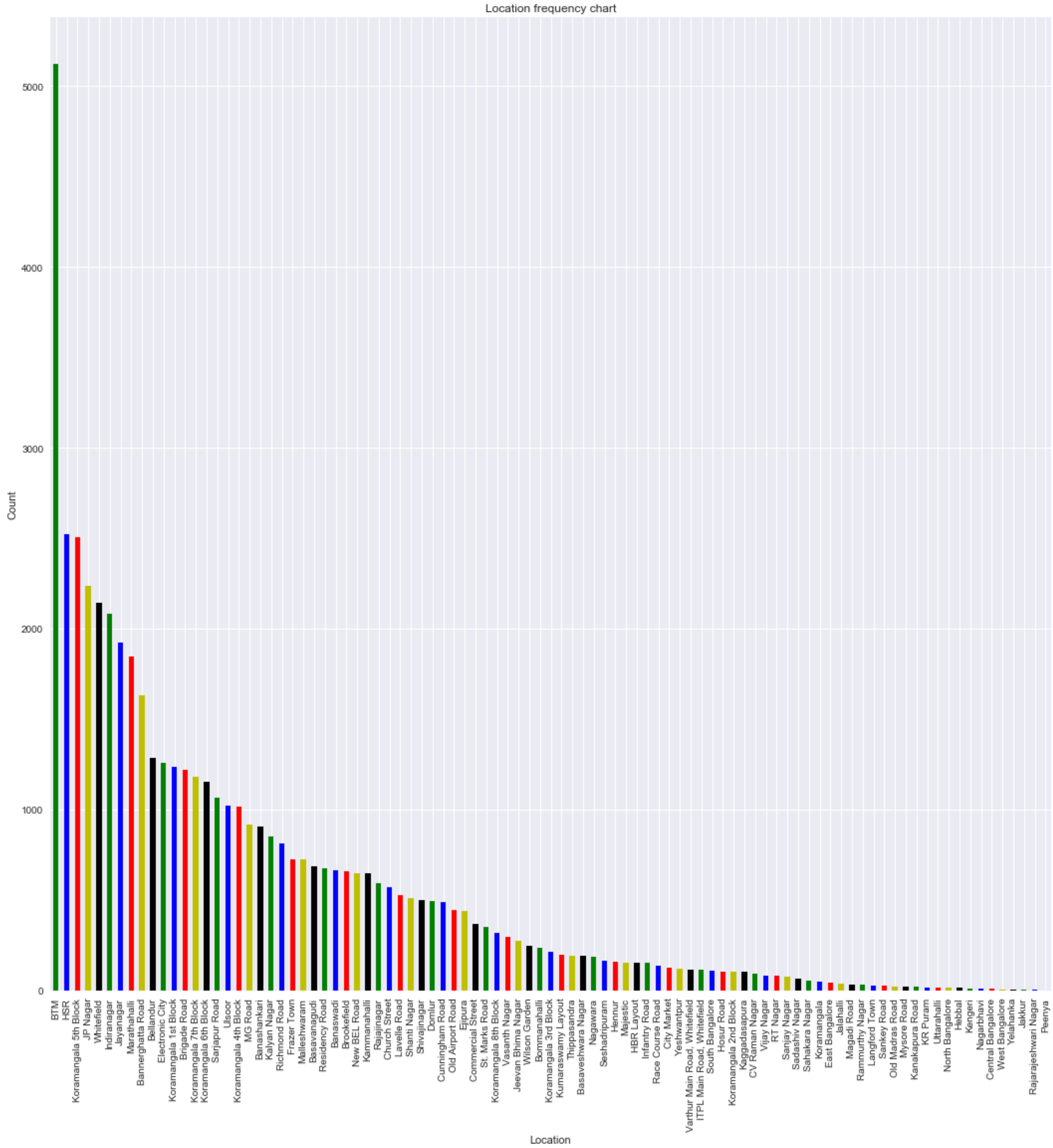
In [0]:

```
plt.hist(df['votes'],bins=15)
plt.title("Histogram of votes")
plt.xlabel('Number of votes')
plt.ylabel('Number of Restaurant')
plt.show()
```



In [0]:

```
df['location'].value_counts().plot(kind='bar',title="Location frequency chart",figsize=
(18,18),color=tuple(["g", "b","r","y","k"]))
plt.xlabel('Location')
plt.ylabel('Count')
plt.show()
```

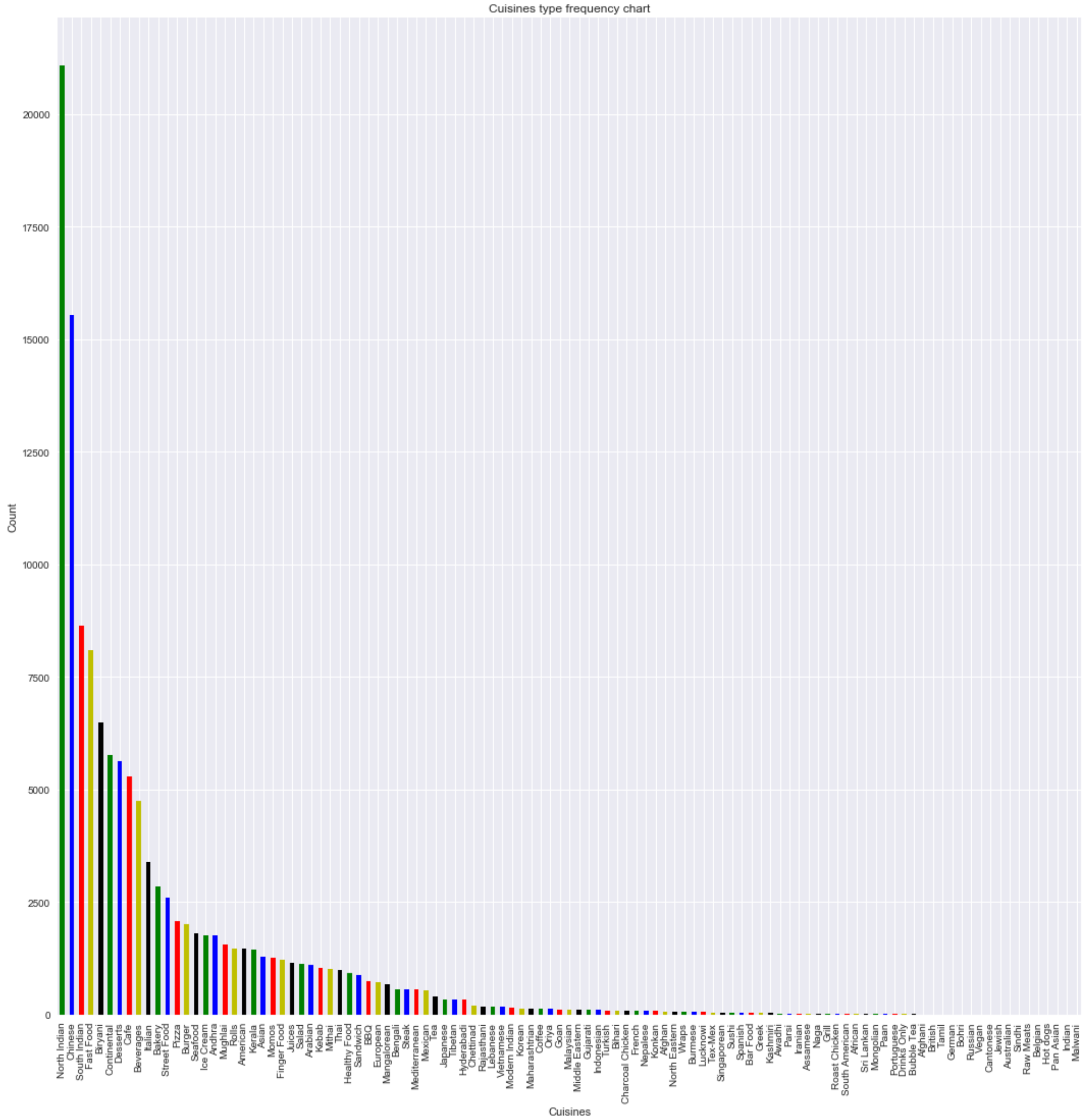

Location frequency chart
Count
5000
4000
3000
2000
1000
0
BTM
HSR
Koramangala 5th Block
JP Nagar
Whitefield
Indiranagar
Jayanagar
Marathahalli
Bannerghatta Road
Bellandur
Electronic City
Koramangala 1st Block
Brigade Road
Koramangala 7th Block
Koramangala 6th Block
Sarjapur Road
Ulsoor
Koramangala 4th Block
MG Road
Banashankari
Kalyan Nagar
Richmond Road
Frazer Town
Malleshwaram
Basavanagudi
Residency Road
Banaswadi
Brookefield
New BEL Road
Kammanahalli
Rajajinagar
Church Street
Lavelle Road
Shanti Nagar
Shivajinagar
Domlur
Cunningham Road
Old Airport Road
Ejipura
Commercial Street
St. Marks Road
Koramangala 8th Block
Vasanth Nagar
Jeevan Bhima Nagar
Wilson Garden
Bommanahalli
Koramangala 3rd Block
Kumaraswamy Layout
Thippasandra
Basaveshwara Nagar
Nagawara
Seshadripuram
Hennur
Majestic
HBR Layout
Infantry Road
Race Course Road
City Market
Yeshwantpur
Varthur Main Road, Whitefield
ITPL Main Road, Whitefield
South Bangalore
Hosur Road
Koramangala 2nd Block
Kaggadasapura
CV Raman Nagar
Vijay Nagar
RT Nagar
Sanjay Nagar
Sadashiv Nagar
Sahakara Nagar
Koramangala
East Bangalore
Jalahalli
Magadi Road
Rammurthy Nagar
Langford Town
Sankey Road
Old Madras Road
Mysore Road
Kanakapura Road
KR Puram
Uttarahalli
North Bangalore
Hebbal
Kengeri
Nagarbhavi
Central Bangalore
West Bangalore
Yelahanka
Jakkur
Rajarajeshwari Nagar
Peenya
Location

cuisines: This attribute displays the cuisines served by the restaurant. There is comma separated values for this attribute. Hence, we have done the split by comma on the attribute and taken all the values in list. After splitting, we have plotted bar graph for this attribute. From the graph we can say that, “North Indian” is served by highest number of the restaurants.
rest_type: It shows the type of the restaurant. This column contains values which are comma separated. For example, value “Beverage Shop, Quick Bites” means that the restaurant server Quick Bites and Beverages both. Hence, similar to “cuisines” attribute, we have done split by comma on this attribute as well. After splitting, we have plotted bar graph for this attribute. From the graph we can say that, “Quick Bites” is the most famous restaurant type.

In [0]:

```
# Display the plot for cusinies
cuisines_column = df['cuisines']
list = []

for cuisines in cuisines_column:
    if(cuisines is None or isinstance(cuisines, float)):
        continue
    else:
        for cuisine in cuisines.split(", "):
            list.append(cuisine)

cuisines= pd.Series(list)
print(cuisines.value_counts().plot(kind='bar',title="Cuisines type frequency chart",col
or=tuple(["g", "b","r","y","k"]),figsize=(18,18)))
plt.xlabel('Cuisines')
plt.ylabel('Count')
plt.show()
```

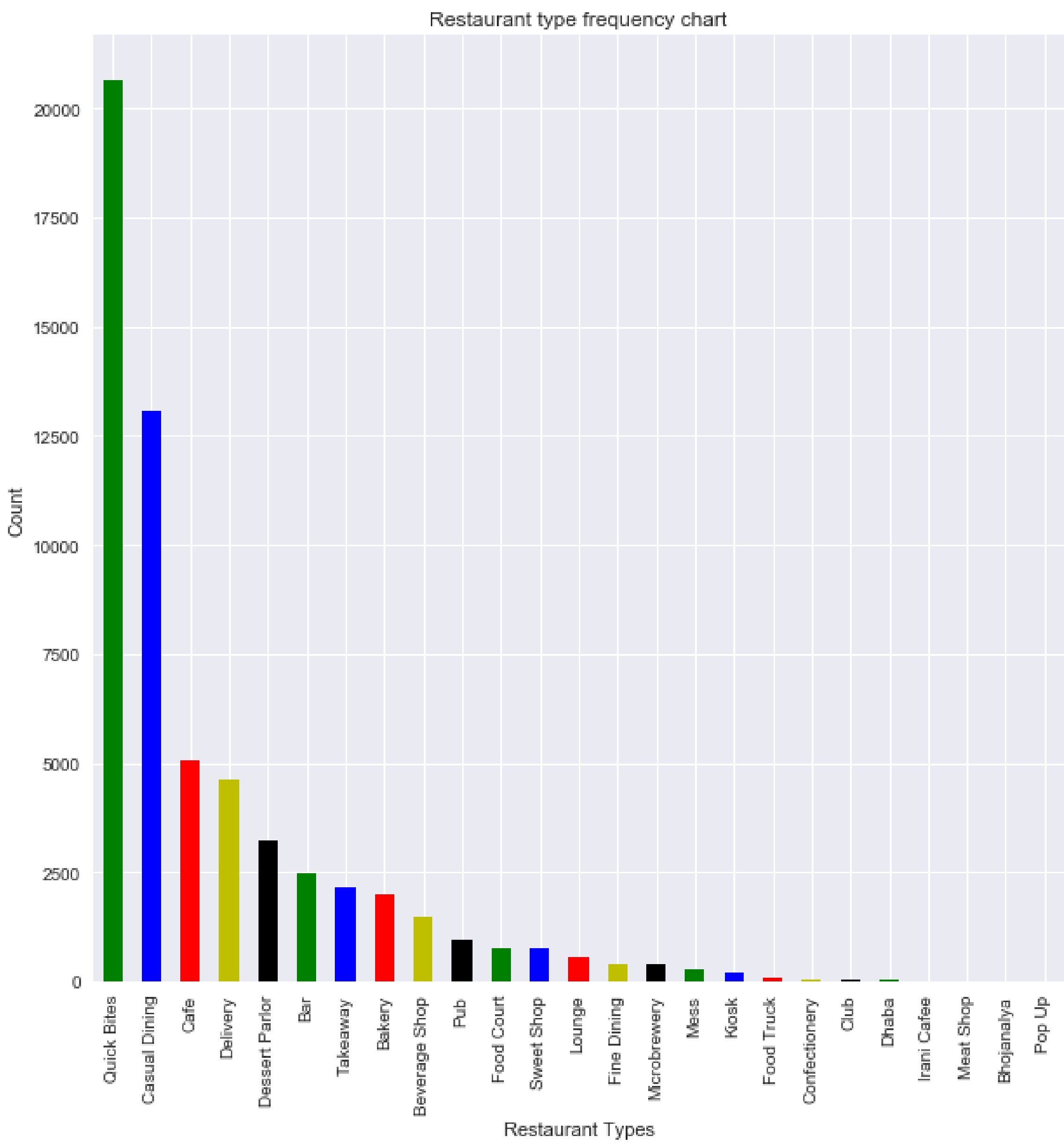
```
AxesSubplot(0.125,0.125;0.775x0.755)
```


Cuisines type frequency chart
Count
20000
17500
15000
12500
10000
7500
5000
2500
0
North Indian
Chinese
South Indian
Fast Food
Biryani
Continental
Desserts
Cafe
Beverages
Italian
Bakery
Street Food
Pizza
Burger
Seafood
Ice Cream
Andhra
Mughlai
Rolls
American
Kerala
Asian
Momos
Finger Food
Juices
Salad
Arabian
Kebab
Mithai
Thai
Healthy Food
Sandwich
BBQ
European
Mangalorean
Bengali
Steak
Mediterranean
Mexican
Tea
Japanese
Tibetan
Hyderabadi
Chettinad
Rajasthani
Lebanese
Vietnamese
Modern Indian
Korean
Maharashtrian
Coffee
Oriya
Goan
Malaysian
Middle Eastern
Gujarati
Indonesian
Turkish
Bihari
Charcoal Chicken
French
Nepalese
Konkan
Afghan
North Eastern
Wraps
Burmese
Lucknowi
Tex-Mex
Singaporean
Sushi
Spanish
Bar Food
Greek
Kashmiri
Awadhi
Parsi
Iranian
Assamese
Naga
Grill
Roast Chicken
South American
African
Sri Lankan
Mongolian
Paan
Portuguese
Drinks Only
Bubble Tea
Afghani
British
Tamil
German
Bohri
Russian
Vegan
Cantonese
Jewish
Australian
Sindhi
Raw Meats
Belgian
Hot dogs
Pan Asian
Indian
Malwani
Cuisines

In [0]:

```
# Display the plot for restaurant types
list = []
rest_type_column = df['rest_type']
for  rest_types in rest_type_column:
    if(rest_types is None or isinstance(rest_types, float)):
        continue
    else:
        for rest_type in rest_types.split(", "):
            list.append(rest_type)

rest_types= pd.Series(list)

print(rest_types.value_counts().plot(kind='bar',title="Restaurant type frequency chart"
,color=tuple(["g", "b","r","y","k"]),figsize=(10,10)))
plt.xlabel('Restaurant Types')
plt.ylabel('Count')
plt.show()
```

AxesSubplot(0.125,0.125;0.775x0.755)



rate: It shows the rating of the restaurant out of 5 stars. It has some values as “NEW”, “-” and null, which means that the restaurant is not rated yet. We have cleaned the data for plotting of the graph. The value for this attribute is a continuous value between 0 to 5. So, we have plotted histogram for this attribute. From the graph, we can say that number of restaurants having rating between 3.5 to 4 is highest.
approx_cost: It shows the approximate cost for two people at the restaurant. The value is continuous value between 0 to 6000. So, we have plotted histogram for this attribute. From the graph, we can say that most of the restaurants has approximate cost between 100 to 800.

Cleaning the "rate" and "approx_cost" column

For cleaning the data in the rate column, we have followed below steps:

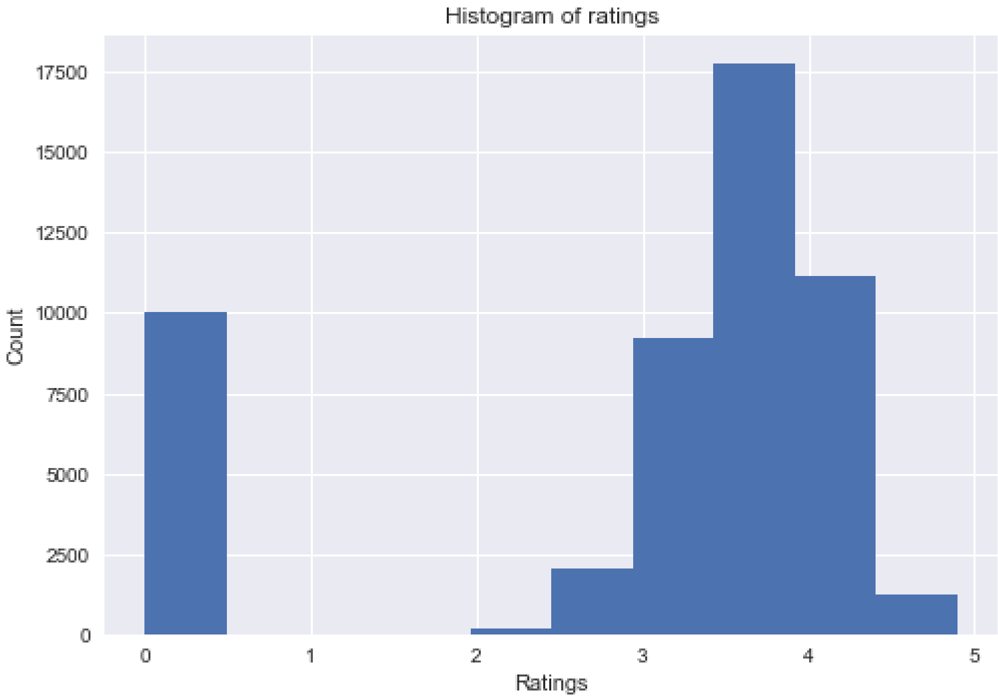
- Create new dataframe df2 as a copy of df
- Replace "NEW" with 0
- Replace "-" with 0
- Fill NA with .
- Converting to string
- Replacing "/5" with null
- Converting the final rate value to float

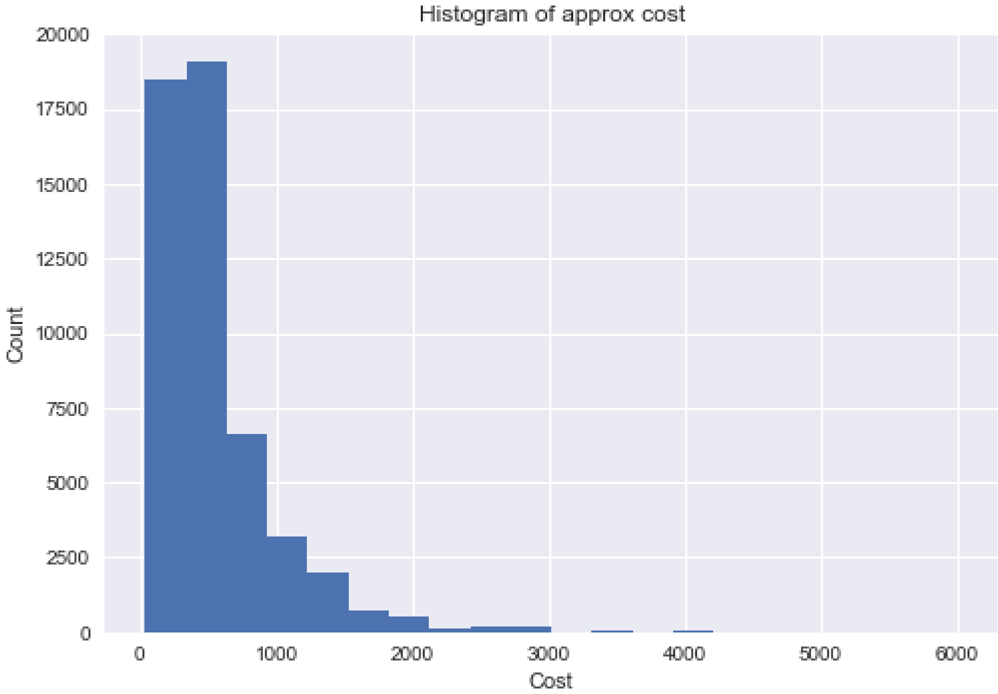
For cleaning the data in the "approx_cost" column, we have followed below steps:

- Replace comma (",") with empty string
- Convert it to float type
- Fill NA with the mean value of "approx_cost"

In [0]:

```
# creating new dataframe and cleaning the ratings cloumn to convert str to float
# This is how conversion is carried out e.g 4.1/5 to 4.1
# All the steps for cleaning the rate columns are written above.
# Reference - https://www.kaggle.com/ranganadhkodali/bangalore-restaurants-eda-analysis
# Reference - https://www.kaggle.com/subbuvolvosekar/best-place-to-eat-at-bangalore
df2=df.copy()
df2['rate'] = df2['rate'].replace('NEW',0)
df2['rate'] = df2['rate'].replace('-',0)
df2['rate']=df2['rate'].fillna(0)
df2['rate'] = df2.loc[:,'rate'].replace('[ ]','',regex = True)
df2['rate'] = df2['rate'].astype(str)
df2['rate'] = df2['rate'].apply(lambda r: r.replace('/5',''))
df2['rate'] = df2['rate'].apply(lambda r: float(r))


# creating plot of ratings
df2['rate'].hist()
plt.title('Histogram of ratings')
plt.xlabel('Ratings')
plt.ylabel('Count')
plt.show()

# drop rows with nan values for mentioned attributes
df2.dropna(subset=["location"],inplace=True)
df2.dropna(subset=["approx_cost(for two people)"],inplace=True)
df2.dropna(subset=["rest_type"],inplace=True)
df2.dropna(subset=["cuisines"],inplace=True)

# cleaning the approx cost and reomve comma and converting into integer and also removi
ng null rows
# e.g "1,200" to 1200
df2['approx_cost(for two people)'] = df2['approx_cost(for two people)'].str.replace(','
,'')
df2['approx_cost(for two people)'] = df2['approx_cost(for two people)'].astype(float)
df2['approx_cost(for two people)'].fillna((df2['approx_cost(for two people)'].mean()),i
nplace=True)

#  creating plot of approx cost
df2['approx_cost(for two people)'].hist(bins=20)
plt.title('Histogram of approx cost')
plt.xlabel('Cost')
plt.ylabel('Count')
plt.show()
```


Histogram of ratings
17500
15000
12500
10000
7500
5000
2500
0
Count
0
1
2
3
4
5
Ratings


Histogram of approx cost
Count
20000
17500
15000
12500
10000
7500
5000
2500
0
0
1000
2000
3000
4000
5000
6000
Cost

Not Useful Attributes

Following are the 9 of attributes out of 17 attributes which are useless at this point.
url: URL will be different for each of the restaurant. And one can't derive any specific information from the url.
address: address will be different for different restaurants. Also, the building numbers and street names can't provide any substantial information. For this purpose, neighbourhood is used.
name: It shows the name of the restaurant. We can't predict approximate cost for 2 people from name of the restaurant.
phone: It contains contact number for the restaurant. Phone number can't help us to predict anything, because it will be different for different restaurants.
dish_liked: It shows the dishes liked by the people for the restaurant. People may like some dish and may not like some other dish. We can't predict the approximate cost from the dished liked for that restaurant.
reviews_list: It contains the reviews of the restaurant. Review is not a useful information in this dataset. We are already using "rate" column for the ratings of the restaurant.
menu_item: It shows the menu items of the restaurant. This attribute can't help in identifying the approximate cost, as some restaurant might have many items, but the cost is less. And vice-versa is also applicable.
listed_in(type): It shows the type of meal. It doesn't provide any significant information. We are using "cuisines" and "rest_type" attributes to identify the food offered by the restaurant.
listed_in(city): It shows the neighbourhood in which the restaurant is located. We are using the "location" attribute, which has almost the same values as this column.

Answer 2-c :- Removing duplicates restaurants

Yes, there are duplicate restaurants in the Zomato data. After cleaning the duplicates, we have got **12382 rows**. For cleaning purpose, we have taken "name" and "address" attributes' unique values. The name of data frame is "df_cleaned".

In [0]:

```
# Removing duplicate restaurant with and removing columns that are not required
# df_cleaned=df2.drop_duplicates(subset =['name','address'], keep = 'first')
df_cleaned=df2.sort_values('votes', ascending=False).drop_duplicates(subset =['name','a
ddress'],keep='first')
print(df_cleaned.shape)
df_cleaned.head(10)
```

```
(12382, 17)
```

Out[0]:

	url	address	name	online_order	book_t
49627	https://www.zomato.com/bangalore/byg-brewski-b...	Behind MK Retail, Sarjapur Road, Bangalore	Byg Brewski Brewing Company	Yes	
18643	https://www.zomato.com/bangalore/toit-indirana...	298, Namma Metro Pillar 62, 100 Feet Road, Ind...	Toit	No	
36668	https://www.zomato.com/bangalore/truffles-kora...	28, 4th 'B' Cross, Koramangala 5th Block, Bang...	Truffles	No	
41525	https://www.zomato.com/bangalore/abs-absolute-...	90/4, 3rd Floor, Outer Ring Road, Munnekollaly...	AB's - Absolute Barbecues	No	
37606	https://www.zomato.com/bangalore/the-black-pea...	105, 1st A Cross Road, Jyothi Nivas College Ro...	The Black Pearl	No	
45609	https://www.zomato.com/bangalore/big-pitcher-a...	LR Arcade,4121, Old Airport Road, Bangalore	Big Pitcher	No	
36690	https://www.zomato.com/bangalore/onesta-korama...	562, 8th Main, Koramangala 4th Block, Bangalore	Onesta	Yes	
48163	https://www.zomato.com/ArborBrewIndia?context=...	8, 3rd Floor, Allied Grande Plaza, Diagonally ...	Arbor Brewing Company	No	
45837	https://www.zomato.com/bangalore/empire-restau...	Next to BSNL, HAL 2nd Stage, 80 Feet Road, Ind...	Empire Restaurant	Yes	
37611	https://www.zomato.com/bangalore/prost-brew-pu...	749, 10th Main, 80 Feet Road, Koramangala 4th ...	Prost Brew Pub	No	

Answer 2-d :- Finding neighbourhood with max average rating

Neighbourhood with the highest average rating is “Lavelle Road”. The average rating for this neighbourhood is **4.07/5**. Most the restaurants in this neighbourhood have approximate cost for two people above **Rs.1200**. Other major characteristics of this neighbourhood is shown in the table below.

In [0]:

```
# Find the neighbourhood of highest average rating and the characteristics of neighbour
hood
df_cleaned['rate'] = df_cleaned['rate'].replace(0,np.nan)
max_average_rating=df_cleaned.groupby('location')['rate'].mean().sort_values(axis=0,asc
ending=False).reset_index()['rate'][0]
print("Max average rating: ",max_average_rating)
location=df_cleaned.groupby('location')['rate'].mean().sort_values(axis=0,ascending=Fal
se).reset_index()['location'][0]
print("Location with max average rating: ",location)
df_neighbourhood=df_cleaned[df_cleaned['location']==location]
print("\nCharacteristics of the location with max average rating ")
df_neighbourhood
```

```
Max average rating:  4.0769230769230775
Location with max average rating:  Lavelle Road

Characteristics of the location with max average rating
```

Out[0]:

	url	address
48783	https://www.zomato.com/bangalore/soda-bottle-o...	25/4, Opposite Harley Davidson Showroom, Lavel...
48188	https://www.zomato.com/bangalore/the-biere-clu...	20/2, Vittal Mallya Road, Lavelle Road, Bangalore
48162	https://www.zomato.com/bangalore/skyye-lavelle...	Uber Level, 16th Floor, UB City, Vittal Mallya...
47413	https://www.zomato.com/SmokeHouseDeli-LavelleR...	52/ 53, Ground Floor, Lavelle Road, Bangalore
48173	https://www.zomato.com/bangalore/farzi-cafe-la...	202, Level 2, UB City, Vittal Mallya Road, Lav...
48775	https://www.zomato.com/bangalore/shiro-lavelle...	2nd Floor, UB City Mall, Vittal Mallya Road, L...
48165	https://www.zomato.com/bangalore/jw-kitchen-jw...	JW Marriott, 24/1, Vittal Mallya Road, Lavelle...
48228	https://www.zomato.com/bangalore/cafe-noir-lav...	2nd Floor, UB City, Vittal Mallya Road, Lavell...
47776	https://www.zomato.com/bangalore/fava-lavelle-...	203, 2nd Floor, UB City, 24 Vittal Mallya Road...
6087	https://www.zomato.com/bangalore/toscano-lavel...	2nd Floor, UB City, Vittal Mallya Road, Lavell...
48809	https://www.zomato.com/bangalore/bootlegger-la...	36, Vittal Mallya Road, Lavelle Road, Bangalore

	url	address	
48203	https://www.zomato.com/bangalore/caperberry-la...	203, 2nd Floor, UB City, 24 Vittal Mallya Road...	
43916	https://www.zomato.com/bangalore/sunnys-1-lave...	50, Opposite Loom, Lavelle Road, Bangalore	
48428	https://www.zomato.com/bangalore/glens-bakehou...	24/1, Lavelle Road, Bangalore	
47242	https://www.zomato.com/bangalore/cafe-mangii-l...	204/A, Comet Block, UB City, Vittal Mallya Roa...	
48787	https://www.zomato.com/bangalore/sanchez-lavel...	204, UB City, Vittal Mallya Road, Near Lavelle...	
48774	https://www.zomato.com/bangalore/kaz%C3%A9-1-l...	909 SKAV, 21st floor, Lavelle Road, Bangalore	KazÃ□Â□Ã□Â□Ã□Â□Ã□Â
48430	https://www.zomato.com/bangalore/airlines-hote...	4, Madras Bank Road, Off Lavelle Road, Lavelle...	
12561	https://www.zomato.com/bangalore/kaz%C3%A9-1-l...	909 SKAV, 21st floor, Lavelle Road, Bangalore	KazÃ□Â
47412	https://www.zomato.com/bangalore/rasovara-lave...	Level 2, The Collection, UB City, Vithal Mally...	
48191	https://www.zomato.com/bangalore/spice-terrace...	JW Marriott, 24/1, Vittal Mallya Road, Lavelle...	
48232	https://www.zomato.com/bangalore/sriracha-lave...	204, 2nd Level, 4th Floor, Comet Block, UB Cit...	

	url	address	
6104	https://www.zomato.com/bangalore/alba-jw-marri...	JW Marriott, 24/1, Vittal Mallya Road, Lavelle...	
37730	https://www.zomato.com/bangalore/bengaluru-bak...	JW Marriott Bengaluru, 24/1, Vittal Mallya Roa...	Benga
47505	https://www.zomato.com/bangalore/the-rice-bowl...	40/2, Lavelle Road, Bangalore	
4995	https://www.zomato.com/bangalore/bbqd-global-g...	Level 2, Concorde Block, UB City, Vittal Malya...	
38596	https://www.zomato.com/bangalore/cafe-coffee-d...	23/2, Vittal Mallya Road, Opposite UB City, La...	
4994	https://www.zomato.com/bangalore/the-spice-baz...	40/2, Lavelle Road, Bangalore	
12392	https://www.zomato.com/bangalore/corner-house-...	4, Madras Bank Road, Lavelle Road, Bangalore	
48002	https://www.zomato.com/bangalore/amande-patiss...	2nd Floor, UB City, Vittal Mallya Road, Lavell...	
48415	https://www.zomato.com/bangalore/soul-city-lav...	Oakwood Premier Prestige, UB City, Vittal Mall...	
6097	https://www.zomato.com/bangalore/tree-tops-bar...	Hotel Southern Star, 40/2, Roof Top, Lavelle R...	
43948	https://www.zomato.com/bangalore/matsuri-the-c...	The Chancery Hotel, Lavelle Road, Bangalore	

	url	address
38446	https://www.zomato.com/bangalore/chocolate-dlu...	24, Ground Floor, Canberra Block, The Collecti...
42873	https://www.zomato.com/bangalore/konark-vegeta...	Sree Kanteerava Outdoor Stadium Main Gate, Kas...
12737	https://www.zomato.com/bangalore/south-parade-...	The Chancery Hotel, Lavelle Road, Bangalore
6186	https://www.zomato.com/bangalore/eatwater-lave...	25/5, Near Lamborghini Showroom, Lavelle Road,...
12925	https://www.zomato.com/bangalore/subway-lavell...	2nd Floor, UB City, Vittal Mallya Road, Lavell...
47307	https://www.zomato.com/bangalore/fresh-presser...	4, Good Earth Store, Former Cinnamon Building,...
48313	https://www.zomato.com/bangalore/the-blackboar...	40/2, Sri ML Subbaraju Road, Shantalanagar, Sa...
43083	https://www.zomato.com/bangalore/keventers-lav...	24,Amphitheatre, UB City, Vittal Mallya Road, ...
48819	https://www.zomato.com/bangalore/bar-uno-jw-ma...	JW Marriott Bengaluru, 24/1, Vittal Mallya Roa...
12223	https://www.zomato.com/bangalore/pizza-stop-la...	Airlines Hotel, 4, Madras Bank Road, Lavelle R...
38056	https://www.zomato.com/bangalore/gmt-gelateria...	42 Vittal Mallya Road, Lavelle Road, Bangalore
48459	https://www.zomato.com/bangalore/mathsya-darsh...	KFDC Ltd, Cubbon Park, K.R.Circle, Lavelle Roa...

	url	address
39322	https://www.zomato.com/bangalore/atomic-lab-la...	#25/5, Lavelle Road, Opposite Smoke House Deli...
12095	https://www.zomato.com/bangalore/cafe-coffee-d...	4/1 Walton Road Circle, Lavelle Road, Bangalore
14737	https://www.zomato.com/bangalore/the-milkshake...	4, Madras Bank Road, Lavelle Road, Bangalore
38605	https://www.zomato.com/bangalore/ozaa-lavelle-...	Oakwood Premier Prestige, UB City, Vittal Mall...
43090	https://www.zomato.com/bangalore/popz-kitchen-...	Opposite UB City, Lavelle Road, Bangalore
5862	https://www.zomato.com/bangalore/royce-chocola...	Bengaluru Baking Company, JW Marriott, Bengalu...
48053	https://www.zomato.com/bangalore/batter-splatt...	Kasturba Road, Sampangi Rama Nagar, Near Lavel...
12058	https://www.zomato.com/bangalore/lazzet-lee-la...	Koppa Road, Yellanahalli Village, Begur Road, ...
12033	https://www.zomato.com/bangalore/cake-my-day-l...	67/1A, 4th Cross, Lavelle Road, Bangalore
12029	https://www.zomato.com/bangalore/chocolate-phi...	4, Walton Road, Lavelle Road, Bangalore
48825	https://www.zomato.com/bangalore/manhattan-1-l...	40/2, Lavelle Road, Bangalore

Attributes Considered

For creating the model, we have used following 5 attributes:

1. location
2. cuisines
3. rest_type
4. rate
5. approx_cost

Out of which, “approx_cost” is the target attribute and rest of the attributes are the features.

In [0]:

```
# consider attributes like location, rest_type, cusisines, rate and approx cost

# df_cleaned['rate'] = df_cleaned['rate'].replace(np.nan,df_cleaned['rate'].mean())
df_cleaned['rate'] = df_cleaned['rate'].replace(np.nan,0)
# df_cleaned.dropna(how='any',inplace=True)
df_data=df_cleaned[['location','rest_type','cuisines','rate','approx_cost(for two peopl
e)']]
df_data.columns=['location','restaurant_type','cuisines','rate','approx_cost']
print(df_data.shape)
df_data.head()
```

```
(12382, 5)
```

Out[0]:

	location	restaurant_type	cuisines	rate	approx_cost
49627	Sarjapur Road	Microbrewery	Continental, North Indian, Italian, South Indi...	4.9	1600.0
18643	Indiranagar	Microbrewery	Italian, American, Pizza	4.7	1500.0
36668	Koramangala 5th Block	Cafe, Casual Dining	Cafe, American, Burger, Steak	4.7	900.0
41525	Marathahalli	Casual Dining	European, Mediterranean, North Indian, BBQ	4.8	1600.0
37606	Koramangala 5th Block	Casual Dining, Bar	North Indian, European, Mediterranean	4.7	1400.0

Answer 3-a :- Identifying the type of problem to be solved

Task

There are approximately 12000 restaurants in Bangalore. Therefore, it becomes difficult for someone to decide which restaurant is better and which restaurant provides better rates. Also, if anyone wants to open a new restaurant in any location, then they have tough competition deciding in which cuisine to choose and what should be the range of approximate cost. Therefore, this Zomato dataset aim to predict the approximate cost for 2 people based on certain number of attributes.

Supervised/Unsupervised

This is supervised learning problem. Because, we have data on target for this dataset. The results in this case will be used to predict the approxmiate cost for any new restaurant based on certain number of attributes. Also, there is a specific purpose — to find the approximate cost.

Hence, it is a **Supervised** problem.

Classification/Regression

This is a regression problem. Becuase, approximate cost for two people is not a categorical value. It is a numeric value. So, the task has a numeric traget — approximate cost for two people.
Hence, this is a **Regression** problem.

Answer 3-b

Choosing Model

We will using following 3 models: Decision Tree, Random Forest, and Xgboost models. Reasons for using this models are discussed below:

Decision Tree

It is one of the most effective models and one of the most popular data mining tools. Decision trees are very simple to understand. It is also easy to implement. Also, most of the data mining packages include support for decision trees. Decision trees performs better on non-linear data. Decision tree regressor will break down the dataset in smaller subset, also simultaneously it will build the associated decision tree. The final tree contains two type of nodes: decision nodes and leaf nodes. A decision node will have two or more branches based on attribute value on which it is divided. Leaf nodes are the target label containing output numeric value. There are many advantages of using decision tree. The final results of decision tree are easy to understand, and it can be summarised using a few IF-THEN conditions. Tree methods are particularly well suited for data where we want to predict the value of target variable based on simple relationship between attributes. In case of our Zomato dataset, we want to find out the value of “approx_cost” for based on the value of 4 attributes: “location”, “rest_type”, “cuisines”, and “rate”. Therefore, we can use decision tree in this case for simple output results.

Reference [https://www.saedsayad.com/decision_tree_reg.htm (https://www.saedsayad.com/decision_tree_reg.htm)](https://www.saedsayad.com/decision_tree_reg.htm)

Reference [http://www.statsoft.com/Textbook/Classification-and-Regression-Trees (http://www.statsoft.com/Textbook/Classification-and-Regression-Trees)](http://www.statsoft.com/Textbook/Classification-and-Regression-Trees)

Random Forest

Random forest is a type of ensemble classifier which is made up of many decision trees. It gives the value of target variable based on most voted value by decision trees. This classifier combines the idea of bagging and random selection of features. In general cases, CART trees are used for creating Random Forest model. Random forest will output the prediciton as the avrage response from all the trees. Hence, in case of our Zomato dataset, it will take average value of "apporx_cost" generated by the number of estimators. For example, we have taken number of estimator as 100. Hence, the final predicted output value will be the average of result generate by these 100 trees. Random forest classifier will increase the accuracy, because we are not relying on output of a single tree and combining the multiple outputs. Therefore, even if one tree makes a large output error, it will be corrected by taking the average of rest 99 trees.

Xgboost

Xgboost algorithm has recently started gaining popularity as an applied machine learning algorithm. It is an implementation of gradient boosted decision trees aiming to improve the performance and speed both. Xgboost is very fast as compared to other gradient based boosting techniques. It is a approach where new models are created after every round that focuses on improving the residual error in last round. Xgboost supports both regression and classification. In our example, we will be using it for regression. It focuses on system optimization using parallelization, tree pruning and hardware optimization. The algorithm uses regularization — which means it will penalize complex algorithms based on LASSO and Ridge regularization to prevent overfitting. In case of our Zomato dataset, we have used Xgboost regressor with estimators, alpha, and 4 other parameters. In this case, our standard deviation was approximately 0.02 — which is better as compared to decision trees and random forest algorithms.

Reference [https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/ (https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/)](https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/)

Importing all the required libraries

In [0]:

```
import sklearn
from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor

from sklearn.model_selection import cross_val_score,KFold
from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error
from sklearn.model_selection import learning_curve
from sklearn.preprocessing import LabelEncoder, OneHotEncoder,MultiLabelBinarizer
from sklearn.model_selection import GridSearchCV

import xgboost as xgb
import warnings
warnings.filterwarnings('ignore')
```

Encoding the data

Two attrbiutes "restaurant_type" and "cuisines" have comma separeted values. So, we have to split those columns. We have applied **one hot encoding** for "location" attribute. After that, we have applied **Multi Lable Binarizer** function to perform one hot encoding on multi-values attributes "restaurant_type" and "cuisines".

In [0]:

```
features=df_data[['location','restaurant_type','cuisines','rate']]
target=df_data['approx_cost']

features['restaurant_type']=features.iloc[:,1].str.split(", ")
features['cuisines']=features.iloc[:,2].str.split(", ")

# one hot encode location using pandas get dummies
features=pd.concat([features, pd.get_dummies(features['location'],prefix='loc',prefix_s
ep='_')], axis=1)

# label_en=LabelEncoder()
# features["location"]=label_en.fit_transform(features["location"])
features.head()
```

Out[0]:

	location	restaurant_type	cuisines	rate	loc_BTM	loc_Banashankari	loc_Ba
49627	Sarjapur Road	[Microbrewery]	[Continental, North Indian, Italian, South Ind...	4.9	0	0	
18643	Indiranagar	[Microbrewery]	[Italian, American, Pizza]	4.7	0	0	
36668	Koramangala 5th Block	[Cafe, Casual Dining]	[Cafe, American, Burger, Steak]	4.7	0	0	
41525	Marathahalli	[Casual Dining]	[European, Mediterranean, North Indian, BBQ]	4.8	0	0	
37606	Koramangala 5th Block	[Casual Dining, Bar]	[North Indian, European, Mediterranean]	4.7	0	0	

5 rows × 97 columns

In [0]:

```
# One hot encoding using multilabelbinariser on restaurant types
mlb = MultiLabelBinarizer()
df1=pd.DataFrame(mlb.fit_transform(features['restaurant_type']),columns=mlb.classes_, i
ndex=features.index)
df1.head()
```

Out[0]:

	Bakery	Bar	Beverage Shop	Bhojanalya	Cafe	Casual Dining	Club	Confectionery	Delivery	De P
49627	0	0	0	0	0	0	0	0	0	
18643	0	0	0	0	0	0	0	0	0	
36668	0	0	0	0	1	1	0	0	0	
41525	0	0	0	0	0	1	0	0	0	
37606	0	1	0	0	0	1	0	0	0	

5 rows × 25 columns

In [0]:

```
# One hot encoding using multilabelbinariser on cuisines
mlb = MultiLabelBinarizer()
df2=pd.DataFrame(mlb.fit_transform(features['cuisines']),columns=mlb.classes_, index=fe
atures.index)
df2.head()
```

Out[0]:

	Afghan	Afghani	African	American	Andhra	Arabian	Asian	Assamese	Australian
49627	0	0	0	0	0	0	0	0	0
18643	0	0	0	1	0	0	0	0	0
36668	0	0	0	1	0	0	0	0	0
41525	0	0	0	0	0	0	0	0	0
37606	0	0	0	0	0	0	0	0	0

5 rows × 106 columns

Final feature set

Attaching the one hot encoded columns and dropping the original columns from the dataframe. Now, all the attributes will have numeric values in the dataset. The dataset will now have 225 columns in total. Top 5 rows of the dataset are shown in in below table:

In [0]:

```
# creating endoded dataframe of the features
features=pd.concat([features,df1,df2],axis=1)
features.drop(['location','cuisines','restaurant_type'],axis=1,inplace=True)
features.head()
```

Out[0]:

	rate	loc_BTM	loc_Banashankari	loc_Banaswadi	loc_Bannerghatta Road	loc_Basavanagud
49627	4.9	0	0	0	0	(
18643	4.7	0	0	0	0	(
36668	4.7	0	0	0	0	(
41525	4.8	0	0	0	0	(
37606	4.7	0	0	0	0	(

5 rows × 225 columns

Creating feature and target set. Then, dividing the features and target variables into training set and test set.

In [0]:

```
# Generating feature and target vector
features=features.values
target=target.values
```

In [0]:

```
# split data in training and testing
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, ra
ndom_state=42)
# List to score evaluation score of each model
Evaluation_score=[0,0,0]
Cross_val_score=[0,0,0]
Parameter_tuning_score=[0,0,0]
```

Decision Tree Regressor

In [0]:

```
# Create Model and train the model. used pre pruning
decisionTree_regressor=DecisionTreeRegressor(random_state=42,max_depth=8,min_impurity_d
ecrease=100)
decisionTree_regressor.fit(X_train,y_train)
target_predict=decisionTree_regressor.predict(X_test)
```

Answer 3-c:- Evaluation Metrics

We have used following metrices to evaluate the performance of models: R2 score, Mean Square Error, Mean Absolute Error.

R2 Score

It is the evaluation matrix which shows how good the data has fit the model. R2 value is always between 0 and 1. 0 means none of the data has fit to the model. 1 means all of the data is completely matched the expected values. Higher the R2 score, better the model performs.

Mean Square Error

Mean square error represents how near a regression decision line is to the data points. It does this by finding the distance of point from the decision lines. Square is required to remove and negative values. The smaller the value of mean squared error, the better the model fit to the data. Depending on the data values, it is possible to get very high MSE or very low MSE. Hence, just by looking at the MSE number one can't tell whether the model is good or not. MSE of testing set will be slightly higher than the MSE of training data.

REF https://www.statisticshowto.datasciencecentral.com/mean-squared-error/ (https://www.statisticshowto.datasciencecentral.com/mean-squared-error/)

Mean Absolute Error

Mean Absolute Error is the most common metrics for measuring the accuracy of the continuous variable. It is the average magnitude of the errors in the predictions. Here, absolute value of the error is necessary, because negative values might cancel out the effect of positive values and we might get very less error, which is wrong representation. If we use Mean Absolute Error, outliers will not play a major role in the prediction. In case of our Zomato dataset, target variable "approx_cost" has continuous numeric value, so we can use Mean Absolute Error.

REF https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d (https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d)

REF https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/mean-absolute-error (https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/mean-absolute-error)

Answer 3-d:- Avoiding overfiting

To prevent over-fitting, we are using pre-pruning in Decision Tree regressor. We have kept maximum depth as 8, which means we are restricting the maximum depth of the decision tree to 8. If we do not specify the max_depth, the tree will grow to full extent. Therefore, we will get approximately 99% accuracy on the training set. However, on the testing dataset, the accuracy will reduce to approximately 60%.
Hence, we are doing pre-pruning in case of decision tree regressor to avoid over-fitting. Hence, the accuracy on training dataset in this case is 78% and accuracy on testing dataset is 71%.

In case of Random Forest Regressor, we are using max_depth and min_impurity_decrease parameters for the pre-pruning of tree. max_depth will ensure that the tree doesn't grow beyond certain depth and min_impurity decrease will make sure that it will do split only if the impurity is decreased by the spcified amount.

In case of Xgboost Regressor, learning_rate is used as step size shrinkage to prevent over-fitting. Max_depth specifying the maximum depth. Alpha shows L1 regularization norm - which mean if we increase its value, the model will be more conservative. Subsample is the ratio for training instances. If we set it to 0.5, then it will sample half of the data before growing the tree. We have used all the above mentioned attributes in case of Xgboost to avoid over-fitting.

Reference: https://xgboost.readthedocs.io/en/latest/parameter.html (https://xgboost.readthedocs.io/en/latest/parameter.html)

Evaluation of Decision Tree

In [0]:

```
# function to evaluate performance of models using r2_score, mean squared error and mea
n absolute error.

def printAccuracy(regressor,index):
    print("R2 score of training data: ",r2_score(y_train,regressor.predict(X_train)))
    print("R2 score of testing data: ",r2_score(y_test, target_predict))
    Evaluation_score[index]=r2_score(y_test, target_predict)
    print("Mean Square error of training data: ",mean_squared_error(y_train,regressor.p
redict(X_train)))
    print("Mean Sqaure error of testing data: ",mean_squared_error(y_test, target_predi
ct))
    print("Mean absolute error of training data: ",mean_absolute_error(y_train,regresso
r.predict(X_train)))
    print("Mean Absolute error of testing data: ",mean_absolute_error(y_test, target_pr
edict))

print("Evaluation of decision tree regressor\n")
printAccuracy(decisionTree_regressor,index=0)
```

```
Evaluation of decision tree regressor

R2 score of training data:  0.7814130274912054
R2 score of testing data:  0.7168326109656803
Mean Square error of training data:  33573.773867674674
Mean Sqaure error of testing data:  43618.02441939844
Mean absolute error of training data:  131.5992363995941
Mean Absolute error of testing data:  137.1196627729331
```

Answer 3-e

We are using cross-validation for evaluating the model. We have kept the kFold size to 5 for Decision tree. Following are the accuracies we got: 0.65520041 0.69931082 0.68375294 0.71457494, and 0.6662126.

- When we ran the cross validation we got approximately following accuracies: 0.65, 0.69, 0.68, 0.71, and 0.66.
- Hence, **Mean Accuracy** is approximately 0.68
- **Standard Deviation** is approximately 2%.

Therefore, as it can be seen from the accuracies that variance is very less. So, our Decision Tree Regressor model is performing well.

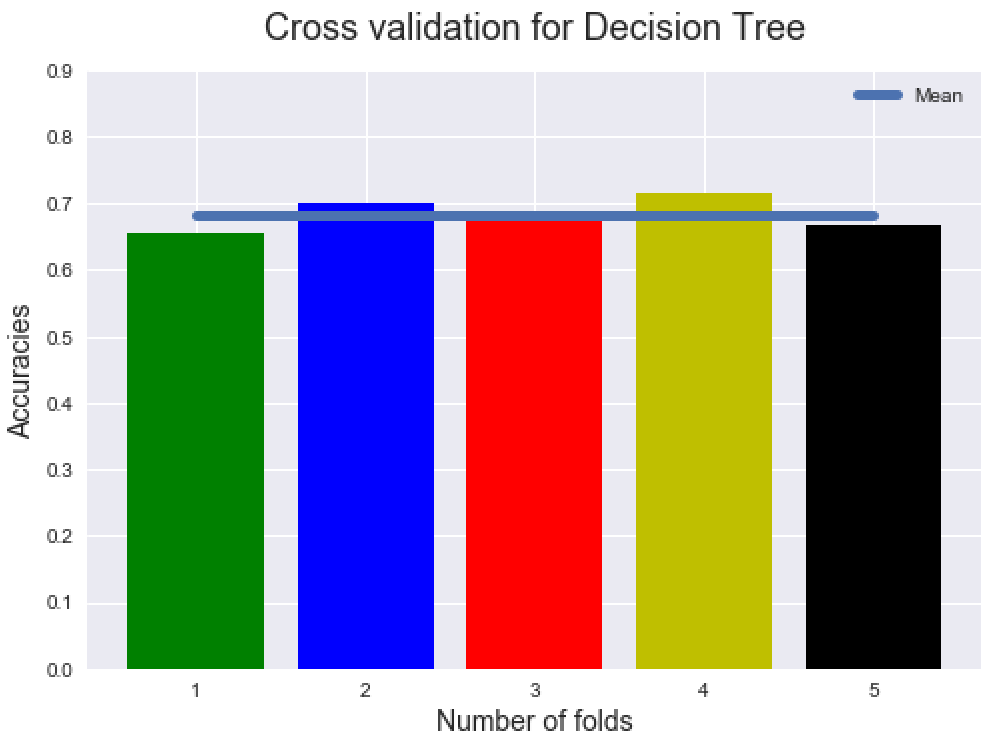
We have also plotted the graph for Cross-Validation for Decision Tree.

Cross-validation for Decision Tree

In [0]:

```
# applying cross validation and finding mean accuracy and standard deviation.
# genarting plot performance for each fold.
def applyCrossValidation(modelName,regressor,kFold,index):
    accuracies=cross_val_score(estimator=regressor,X=X_train,y=y_train,cv=kFold,n_jobs=
-1)
    print("Accuracies",accuracies)
    print("Mean Accuracies",accuracies.mean())
    Cross_val_score[index]=accuracies.mean()
    print("Standard deviation",accuracies.std())
    index=[i for i in range(1,kFold+1)]
    plt.bar(index,accuracies,color=tuple(["g", "b","r","y","k"]))
    plt.plot(index,np.full(kFold,accuracies.mean(),dtype="float"),linewidth=5.0,label=
"Mean")
    plt.xlabel('Number of folds', fontsize=14)
    plt.ylabel('Accuracies', fontsize=14)
    plt.title("Cross validation for "+ modelName, fontsize = 18,y=1.03)
    plt.ylim(0,0.9)
    plt.legend(loc="upper right")
    plt.show()
applyCrossValidation("Decision Tree",decisionTree_regressor,kFold=5,index=0)
```

```
Accuracies [0.65520041 0.69931082 0.68375294 0.71457494 0.6662126 ]
Mean Accuracies 0.6838103404987719
Standard deviation 0.021517143143646725
```



Answer 3-f

We have plotted learning curve to test our model on testing data. It shows the generalization performance plotted against the amount of training data. As shown in the learning curve graph below, mean absolute error for small training size is less for training set and high for testing set. However, as the training set size increases, the mean absolute error for both training set and testing becomes stable as shown in graph below.

REF :- https://www.dataquest.io/blog/learning-curves-machine-learning/ (https://www.dataquest.io/blog/learning-curves-machine-learning/)

Plot Learning Curve for Decision Tree

In [0]:

```
# ploting learning curve for the models
def plotLearningCurve(modelName,regressor,kFold):
    train_sizes = [1, 100, 500, 2000, 7920]
    train_sizes, train_scores, test_scores = learning_curve(
    estimator = regressor,
    X = X_train,
    y = y_train, train_sizes = train_sizes, cv = kFold,
    scoring = 'neg_mean_absolute_error')
    train_scores_mean = -train_scores.mean(axis = 1)
    test_scores_mean = -test_scores.mean(axis = 1)
    plt.style.use('seaborn')
    plt.plot(train_sizes, train_scores_mean, label = 'Training error',linewidth=3)
    plt.plot(train_sizes, test_scores_mean, label = 'Test error',linewidth=3)
    plt.ylabel('Mean absolute error', fontsize = 14)
    plt.xlabel('Training set size', fontsize = 14)
    plt.title("Learning curves for "+ modelName, fontsize = 18,y=1.05)
    plt.legend()
    plt.ylim(0,350)
plotLearningCurve("Decision Tree Regression Model",decisionTree_regressor,kFold=5)
```


Learning curves for Decision Tree Regression Model
Training error
Test error
Mean absolute error
0
50
100
150
200
250
300
350
0
1000
2000
3000
4000
5000
6000
7000
8000
Training set size

Answer 3-g

We are using Grid Search Parameter Tuning for tuning our model. In grid search parameter, we can pass number of parameter and it will perform the modelling using combination of each of those parameters. Also, we can specify various numbers of kFold. Once the grid search cv is performed, it will return the best combination of the parameter as well as the best k fold number.

Following are the attributes used for grid search parameter tuning:

- max_depth
- min_impurity_decrease

When we performed the grid search paramter on decision tree regressor, the values are:

- Best Score = 0.7113351265806567
- Best Parameters = {'max_depth': 18, 'min_impurity_decrease': 70}.

NOTE: The values may vary a little when we run the tuning second time.

Reference: [https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html)](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html)

Grid Search Parameter Tuning for Decision Tree

Following are the parameters used for Decision Tree parameter tuning:

- max_depth
- min_impurity_decrease

In [0]:

```
# tuning parameters using gridSearch method and finding best model and best paramaeters

def GridSearchParameterTuning(parameters,regressor,kFold,index):
    grid_search=GridSearchCV(estimator=regressor,
                             param_grid=parameters,
                             cv=kFold,
                             n_jobs=-1)

    # train model using grid search
    grid_search=grid_search.fit(X_train,y_train)

    best_score=grid_search.best_score_
    Parameter_tuning_score[index]=best_score
    best_parameters=grid_search.best_params_
    print(grid_search.best_estimator_)
    print("Best Score",best_score)
    print("Best Parameters",best_parameters)

parameters_decisionTree={'max_depth':[i for i in range(6,20)],"min_impurity_decrease":[
50,60,70,100]}
GridSearchParameterTuning(parameters_decisionTree,decisionTree_regressor,kFold=5,index=
0)
```

```
DecisionTreeRegressor(criterion='mse', max_depth=18, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=70,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=42, splitter='best')
Best Score 0.7113351265806567
Best Parameters {'max_depth': 18, 'min_impurity_decrease': 70}
```

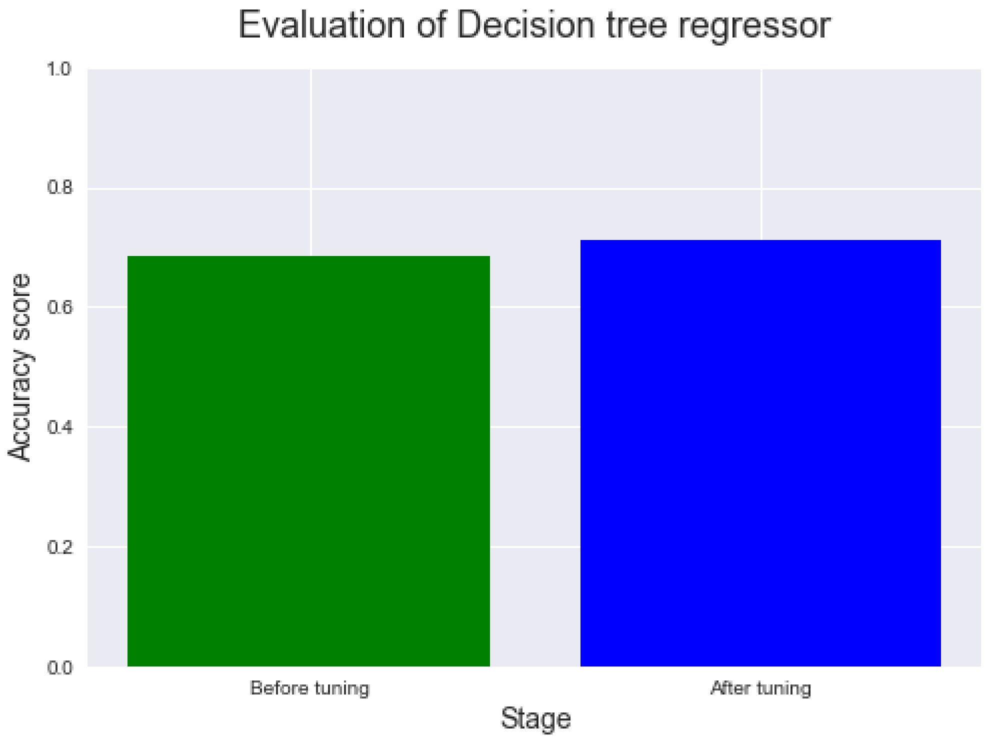
Evaluate the performance of decsion tree regressor after parameter tuning

The bar graph below depicts that the performance of decsion tree regressor improves by some margin after tuning the parameters.

In [0]:

```
def plotAfterParameterTuning(modelName,index):
    print("Score Before tuning: ",Cross_val_score[index])
    print("Score after parameter tuning: ",Parameter_tuning_score[index])
    plt.bar(["Before tuning","After tuning"],[Cross_val_score[index],Parameter_tuning_s
core[index]],color=tuple(["g", "b","r","y","k"]))
    plt.xlabel('Stage', fontsize=14)
    plt.ylabel('Accuracy score', fontsize=14)
    plt.title("Evaluation of "+ modelName, fontsize = 18,y=1.03)
    plt.ylim(0,1)
    plt.show()

plotAfterParameterTuning("Decision tree regressor",0)
```

```
Score Before tuning:  0.6838103404987719
Score after parameter tuning:  0.7113351265806567
```



Random Forest Regressor

In [0]:

```
#  Random forest regressor
RF_regressor=RandomForestRegressor(n_estimators=100,random_state=0,max_depth=12,min_imp
urity_decrease=70)
RF_regressor.fit(X_train,y_train)
target_predict=RF_regressor.predict(X_test)
```

In [0]:

```
print("Evaluation of Random forest regressor \n")
printAccuracy(RF_regressor,index=1)
```

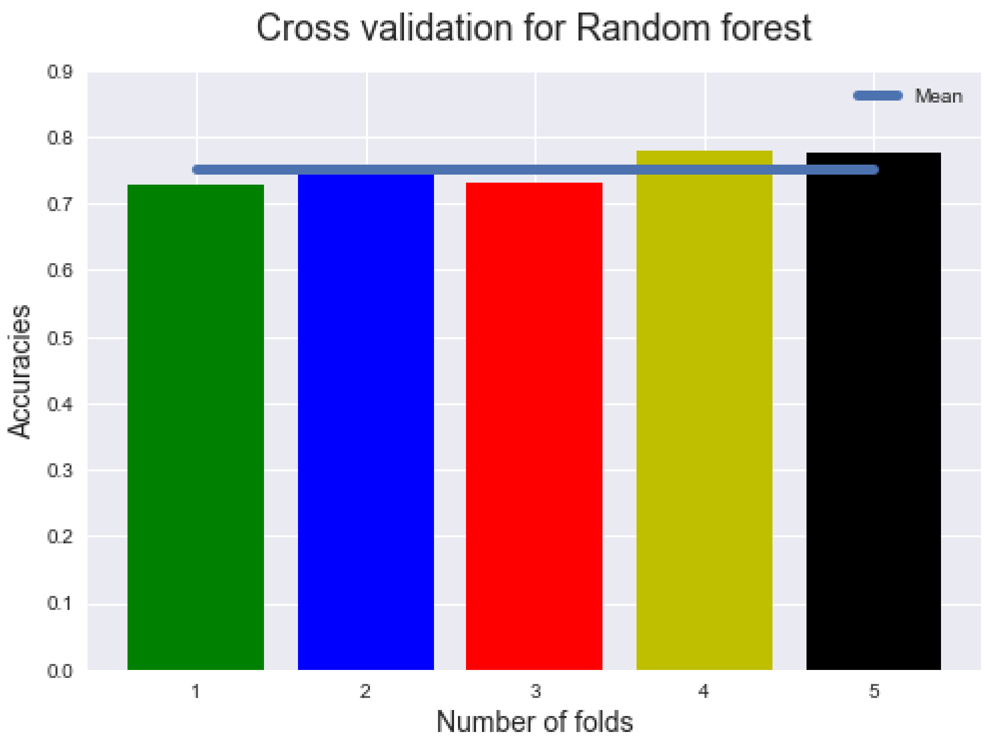
```
Evaluation of Random forest regressor

R2 score of training data:  0.8362117431900468
R2 score of testing data:  0.7630100765266669
Mean Square error of training data:  25156.988237699086
Mean Sqaure error of testing data:  36505.02377573701
Mean absolute error of training data:  118.37969670217441
Mean Absolute error of testing data:  128.34771324765308
```

Cross-validation for Random Forest

In [0]:

```
# applying cross validation and finding mean accuracy and standard deviation
print("Applying cross validation for random forest")
applyCrossValidation("Random forest",RF_regressor,kFold=5,index=1)
```

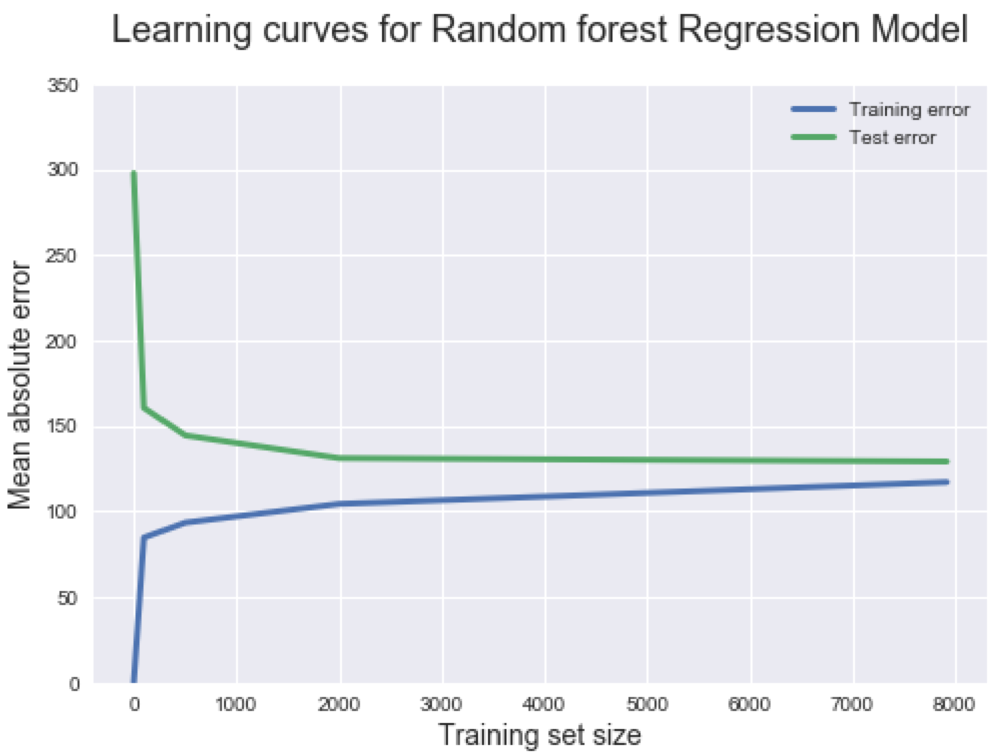
```
Applying cross validation for random forest
Accuracies [0.72794862 0.74154258 0.7309652  0.78047852 0.77614884]
Mean Accuracies 0.7514167496406696
Standard deviation 0.022462327466685451
```



Plot Learning Curve for Random Foresst

In [0]:

```
plotLearningCurve("Random forest Regression Model",RF_regressor,kFold=5)
```



Grid Search Parameter Tuning for Random Forest

Following are the parameters used for Random Forest parameter tuning:

- max_depth
- n_estimators

In [0]:

```
# tuning parameters for random forest
parameters_randomForest=[{'n_estimators':[100,200,300],'max_depth':[i for i in range(8,
20,2)]}]
GridSearchParameterTuning(parameters_randomForest,RF_regressor,kFold=5,index=1)
```

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=18,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=70, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=300, n_jobs=None,
           oob_score=False, random_state=0, verbose=0, warm_start=False)
Best Score 0.7601960820475202
Best Parameters {'max_depth': 18, 'n_estimators': 300}
```

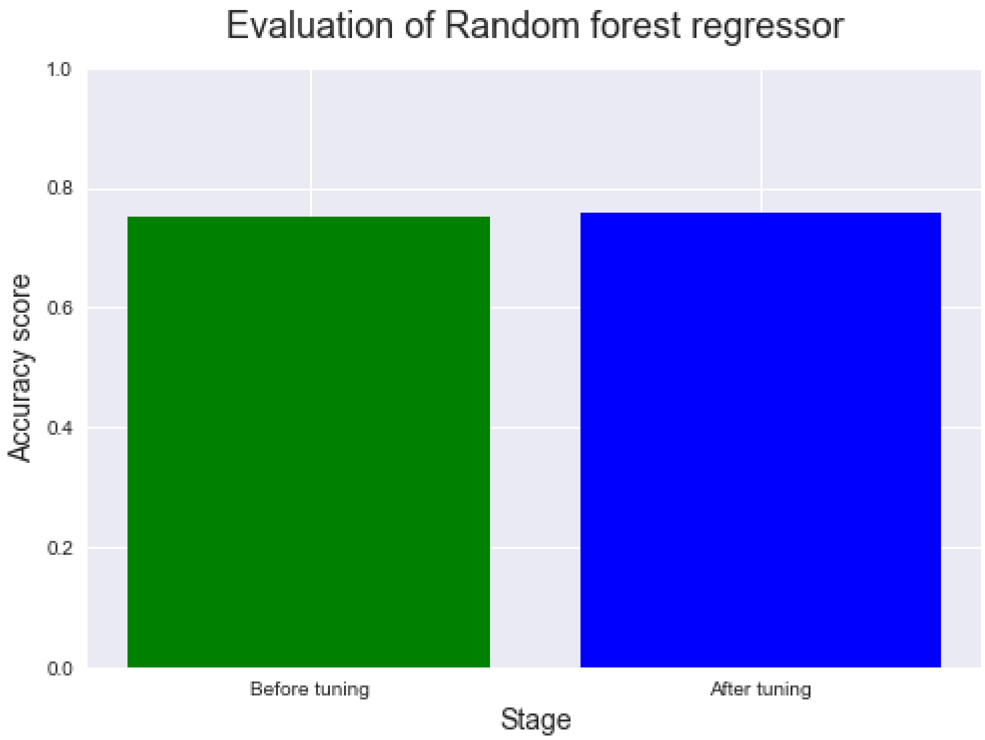
Evaluate the performance of decsion tree regressor after parameter tuning

The bar graph below depicts that the performance of decsion tree regressor improves by some margin after tuning the parameters.

In [0]:

```
plotAfterParameterTuning("Random forest regressor",1)
```

```
Score Before tuning:  0.7514167496406696
Score after parameter tuning:  0.7601960820475202
```



Xgboost Regressor

In [0]:

```
# Xgboost regressor
# Fitting XGB regressor
xgBoost_regressor = xgb.XGBRegressor(colsample_bytree=1,
                 gamma=0,
                 learning_rate=0.07,
                 min_child_weight=1.5,
                 max_depth=5,
                 n_estimators=150,
                 alpha=10,
                 subsample=0.5,
                 seed=42)
# regressor = xgb.XGBRegressor(n_estimators=1000,gamma=0)
xgBoost_regressor.fit(X_train,y_train)
target_predict=xgBoost_regressor.predict(X_test)
```

```
[08:59:30] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
```

Print accuracy for Xgboost

In [0]:

```
print("Evaluation of Xgboost regressor \n")
printAccuracy(xgBoost_regressor,index=2)
```

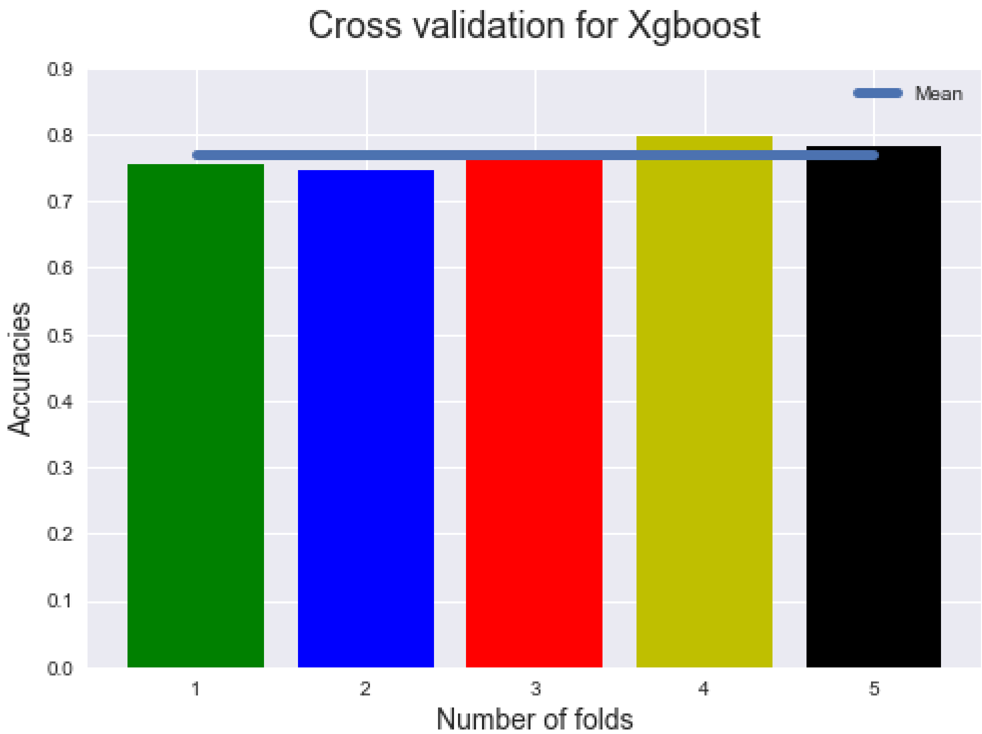
```
Evaluation of Xgboost regressor

R2 score of training data:  0.836048358603892
R2 score of testing data:  0.7962194947480145
Mean Square error of training data:  25182.083224312704
Mean Sqaure error of testing data:  31389.57167557588
Mean absolute error of training data:  112.39055903674014
Mean Absolute error of testing data:  119.96045931530037
```

Cross-validation for Xgboost

In [0]:

```
# applying cross validation and finding mean accuracy and standard deviation for xgboos
t regressorr
print("Applying cross validation for Xgboost regressor")
applyCrossValidation("Xgboost",xgBoost_regressor,kFold=5,index=2)
```

```
Applying cross validation for Xgboost regressor
Accuracies [0.75641934 0.74709841 0.75987454 0.79780197 0.78215428]
Mean Accuracies 0.7686697069185099
Standard deviation 0.018564858902228493
```



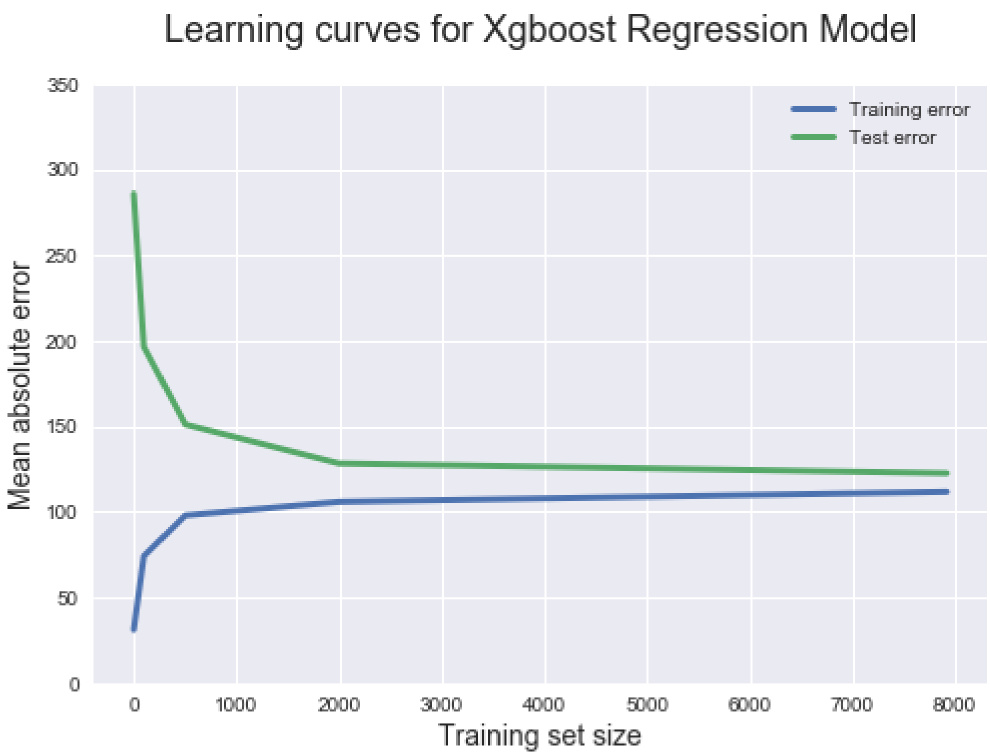
Plot Learning Curve for Xgboost

In [0]:

```
plotLearningCurve("Xgboost Regression Model",xgBoost_regressor,kFold=5)
```

```
[09:00:17] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:00:17] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:00:18] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:00:19] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:00:23] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:00:36] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:00:36] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:00:36] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:00:37] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:00:40] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:00:53] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:00:53] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:00:53] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:00:54] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:00:57] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:01:09] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:01:09] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:01:10] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:01:10] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:01:13] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:01:26] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
```

```
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:01:26] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:01:26] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:01:27] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
[09:01:30] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
```



Grid Search Parameter Tuning for Xgboost

The parameters to be tuned are stored in the dictionary called "parameters_xgboost".This set of paramters are passed to GridSearchCV algorithm.

Following are the parameters used for Xgboost parameter tuning:

- colsample_bytree
- min_child_weight
- learning_rate
- max_depth
- subsample

In [0]:

```
# tuning parameters of xgboost

parameters_xgboost= {
   'colsample_bytree':[0.6,0.8,1],
   'min_child_weight':[1.5,2],
   'learning_rate':[0.1,0.07],
   'max_depth':[5,6],
   'subsample':[0.6,0.5]
}

GridSearchParameterTuning(parameters_xgboost,xgBoost_regressor,kFold=5,index=2)
```

```
[09:24:52] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/ob
jective/regression_obj.cu:152: reg:linear is now deprecated in favor of re
g:squarederror.
XGBRegressor(alpha=10, base_score=0.5, booster='gbtree', colsample_bylevel
=1,
       colsample_bynode=1, colsample_bytree=0.8, gamma=0,
       importance_type='gain', learning_rate=0.07, max_delta_step=0,
       max_depth=6, min_child_weight=1.5, missing=None, n_estimators=150,
       n_jobs=1, nthread=None, objective='reg:linear', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=42, silent=Non
e,
       subsample=0.6, verbosity=1)
Best Score 0.773579179160643
Best Parameters {'colsample_bytree': 0.8, 'learning_rate': 0.07, 'max_dept
h': 6, 'min_child_weight': 1.5, 'subsample': 0.6}
```

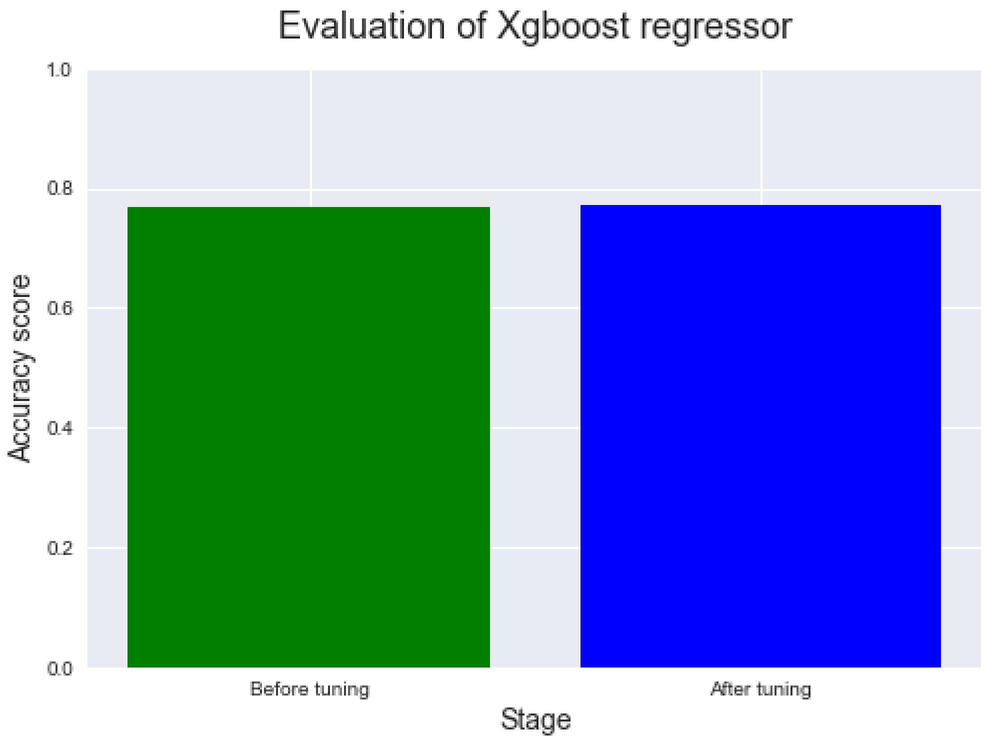
Evaluate the performance of xgboost regressor after parameter tuning

The bar graph below depicts that the performance of Xgboost regressor improves by some margin after tuning the parameters.

In [0]:

```
plotAfterParameterTuning("Xgboost regressor",2)
```

```
Score Before tuning:  0.7686697069185099
Score after parameter tuning:  0.773579179160643
```

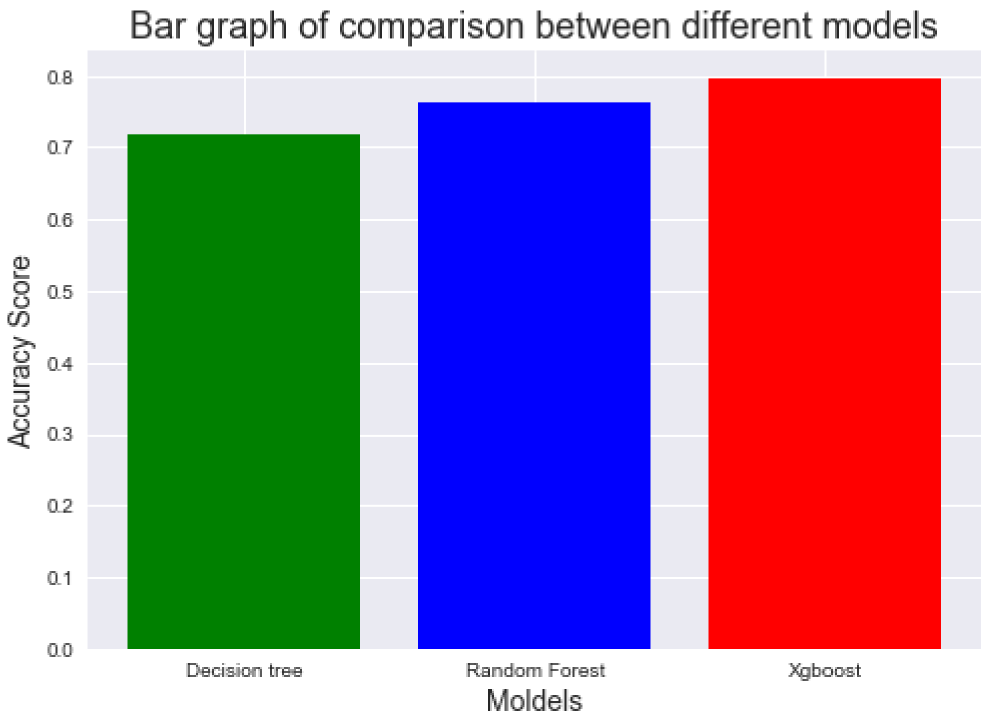


Comparison of all three models

The bar below shows chows that Xgboost algorithm performs better than decision tree and random forest regressor.

In [0]:

```
# comparision performance of the models
plt.bar(['Decision tree','Random Forest','Xgboost'],Evaluation_score,color=tuple(["g",
"b","r","y","k"]))
plt.xlabel('Moldels', fontsize=14)
plt.ylabel('Accuracy Score', fontsize=14)
plt.title("Bar graph of comparison between different models", fontsize = 18)
plt.show()
```



Answer 3-h (BONUS) Relief feature selection

ReliefF algorithms aims to identify predictive features of the model in case of supervised learning. It also identifies the feature interactions which are generally not taken care by the standard algorithms. We have used following parameters form the ReliefF algorithm:

- n_features_to_keep - It specifies the number of features we want finalise for our model.
- n_neighbors - It specifies the number of neighbors to consider while assessing this feature.

Reference: [https://epistasislab.github.io/scikit-rebate/using/ (https://epistasislab.github.io/scikit-rebate/using/)](https://epistasislab.github.io/scikit-rebate/using/)

Reference:- [https://libraries.io/pypi/ReliefF (https://libraries.io/pypi/ReliefF)](https://libraries.io/pypi/ReliefF)

In [0]:

```
!pip install skrebate
!pip install ReliefF==0.1.2
```

```
Requirement already satisfied: skrebate in c:\users\shrey amin\appdata\loc
al\programs\python\python37\lib\site-packages (0.6)
Requirement already satisfied: numpy in c:\users\shrey amin\appdata\local
\programs\python\python37\lib\site-packages (from skrebate) (1.16.0)
Requirement already satisfied: scipy in c:\users\shrey amin\appdata\local
\programs\python\python37\lib\site-packages (from skrebate) (1.2.0)
Requirement already satisfied: scikit-learn in c:\users\shrey amin\appdata
\local\programs\python\python37\lib\site-packages (from skrebate) (0.20.2)

You are using pip version 19.0.2, however version 19.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pi
p' command.

Collecting ReliefF==0.1.2
  Downloading https://files.pythonhosted.org/packages/98/f1/3d8bb05c448b3e
d5e6a436166344b3aafa71848de8f5ee2595489627fc5c/ReliefF-0.1.2.tar.gz (48kB)
Requirement already satisfied: numpy in c:\users\shrey amin\appdata\local
\programs\python\python37\lib\site-packages (from ReliefF==0.1.2) (1.16.0)
Requirement already satisfied: scipy in c:\users\shrey amin\appdata\local
\programs\python\python37\lib\site-packages (from ReliefF==0.1.2) (1.2.0)
Requirement already satisfied: scikit-learn in c:\users\shrey amin\appdata
\local\programs\python\python37\lib\site-packages (from ReliefF==0.1.2)
(0.20.2)
Installing collected packages: ReliefF
  Running setup.py install for ReliefF: started
    Running setup.py install for ReliefF: finished with status 'done'
Successfully installed ReliefF-0.1.2

You are using pip version 19.0.2, however version 19.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pi
p' command.
```

In [0]:

```
from sklearn.pipeline import make_pipeline
from skrebate import ReliefF,SURF
from ReliefF import ReliefF
```

Relief feature selection using ReliefF alogrithm

In [0]:

```
feature_set_2 = ReliefF(n_neighbors=100, n_features_to_keep=2)
X_train_subset = feature_set_2.fit_transform(X_train, y_train)
print(X_train_subset)
print(X_train_subset.shape)
X_test_subset = feature_set_2.transform(X_test)
print(X_test_subset)
print(X_test_subset.shape)
```

```
[[3.8 1. ]
 [3.8 1. ]
 [4.5 0. ]
 ...
 [3.7 1. ]
 [3.9 1. ]
 [3.7 0. ]]
(9905, 2)
[[3.3 0. ]
 [4.2 0. ]
 [3.7 0. ]
 ...
 [0.  0. ]
 [4.1 0. ]
 [3.4 1. ]]
(2477, 2)
```

In [0]:

```
feature_set_4 = ReliefF(n_neighbors=100, n_features_to_keep=4)
X_train_subset = feature_set_4.fit_transform(X_train, y_train)
print(X_train_subset)
print(X_train_subset.shape)
X_test_subset = feature_set_4.transform(X_test)
print(X_test_subset)
print(X_test_subset.shape)
```

```
[[3.8 1.  0.  0. ]
 [3.8 1.  1.  0. ]
 [4.5 0.  0.  0. ]
 ...
 [3.7 1.  0.  0. ]
 [3.9 1.  1.  0. ]
 [3.7 0.  0.  0. ]]
(9905, 4)
[[3.3 0.  0.  1. ]
 [4.2 0.  1.  0. ]
 [3.7 0.  0.  0. ]
 ...
 [0.  0.  1.  0. ]
 [4.1 0.  0.  0. ]
 [3.4 1.  1.  0. ]]
(2477, 4)
```

In [0]:

```
feature_set_6 = ReliefF(n_neighbors=100, n_features_to_keep=6)
X_train_subset = feature_set_6.fit_transform(X_train, y_train)
print(X_train_subset)
print(X_train_subset.shape)
X_test_subset = feature_set_6.transform(X_test)
print(X_test_subset)
print(X_test_subset.shape)
```

```
[[3.8 1.  0.  0.  0.  0. ]
 [3.8 1.  1.  0.  1.  0. ]
 [4.5 0.  0.  0.  0.  0. ]
 ...
 [3.7 1.  0.  0.  0.  0. ]
 [3.9 1.  1.  0.  0.  0. ]
 [3.7 0.  0.  0.  0.  1. ]]
(9905, 6)
[[3.3 0.  0.  1.  0.  0. ]
 [4.2 0.  1.  0.  0.  0. ]
 [3.7 0.  0.  0.  0.  0. ]
 ...
 [0.  0.  1.  0.  0.  1. ]
 [4.1 0.  0.  0.  0.  0. ]
 [3.4 1.  1.  0.  0.  0. ]]
(2477, 6)
```

In [0]:

```
feature_set_8 = ReliefF(n_neighbors=100, n_features_to_keep=8)
X_train_subset = feature_set_8.fit_transform(X_train, y_train)
print(X_train_subset)
print(X_train_subset.shape)
X_test_subset = feature_set_8.transform(X_test)
print(X_test_subset)
print(X_test_subset.shape)
```

```
[[3.8 1.  0.  ... 0.  0.  0. ]
 [3.8 1.  1.  ... 0.  0.  0. ]
 [4.5 0.  0.  ... 0.  0.  0. ]
 ...
 [3.7 1.  0.  ... 0.  0.  0. ]
 [3.9 1.  1.  ... 0.  0.  0. ]
 [3.7 0.  0.  ... 1.  1.  0. ]]
(9905, 8)
[[3.3 0.  0.  ... 0.  1.  0. ]
 [4.2 0.  1.  ... 0.  1.  0. ]
 [3.7 0.  0.  ... 0.  0.  0. ]
 ...
 [0.  0.  1.  ... 1.  1.  0. ]
 [4.1 0.  0.  ... 0.  0.  0. ]
 [3.4 1.  1.  ... 0.  1.  1. ]]
(2477, 8)
```

Relief feature selection using SURF algorithm

In [0]:

```
# Feature selection using the Relief ALgorithm- Different number of features changes ac
curacy that you can check from cross_val_score
# We have used SURF algorithm for relief feature selection
# creating sklearn pipeline and defining number of features to be considered

reg_relief_2 = make_pipeline(SURF(n_features_to_select=2,n_jobs=-1),DecisionTreeRegress
or())
reg_relief_8 = make_pipeline(SURF(n_features_to_select=8,n_jobs=-1),DecisionTreeRegress
or())
reg_relief_4 = make_pipeline(SURF(n_features_to_select=4,n_jobs=-1),DecisionTreeRegress
or())
reg_relief_6 = make_pipeline(SURF(n_features_to_select=6,n_jobs=-1),DecisionTreeRegress
or())
reg_relief_16 = make_pipeline(SURF(n_features_to_select=16,n_jobs=-1),DecisionTreeRegre
ssor())
```

In [0]:

```
cvs_2=cross_val_score(reg_relief_2, features, target)
print('Cross_Value_Score',cvs_2)
print('Mean of cross value score',np.mean(cvs_2))
```

In [0]:

```
cvs_4=cross_val_score(reg_relief_4, features, target)
print('Cross_Value_Score',cvs_2)
print('Mean of cross value score',np.mean(cvs_4))
```

In [0]:

```
cvs_6=cross_val_score(reg_relief_6, features, target)
print('Cross_Value_Score',cvs_2)
print('Mean of cross value score',np.mean(cvs_6))
```

In [0]:

```
cvs_8=cross_val_score(reg_relief_8, features, target)
print('Cross_Value_Score',cvs_2)
print('Mean of cross value score',np.mean(cvs_8))
```

In [0]:

```
cvs_16=cross_val_score(reg_relief_16, features, target)
print('Cross_Value_Score',cvs_2)
print('Mean of cross value score',np.mean(cvs_16))
```