' ---
Title: "EDA on a Food-Inspection Dataset"
Author: "Sumon Singh"
'Date: "Oct19th, 2021"
' ---

Q1.Perform Exploratory Data analysis on the dataset.

Code :

```
df_Food = read.csv('/home/sumon/Downloads/Documents/food-inspections.csv')

View(head(df_Food))
```

Output :

Filter

	Inspection.ID	DBA.Name	AKA.Name	License..	Facility.Type	Risk	Address	City	State	Zip	Inspection.Date	Inspection.Type	Results	Violations
1	2352683	CHIPOTLE MEXICAN GRILL	CHIPOTLE MEXICAN GRILL	2670642	Restaurant	Risk 1 (High)	1025 W ADDISON ST	CHICAGO	IL	60613	2019-12-04T00:00:00.000	License Re-Inspection	Pass w/ Conditions	3. MANAGEMENT, FOOD EMPL
2	2352734	CHILI'S T-I	CHILI'S (T1-B14)	34169	Restaurant	Risk 1 (High)	11601 W TOUHY AVE	CHICAGO	IL	60666	2019-12-04T00:00:00.000	Canvass	Pass	10. ADEQUATE HANDWASHIN
3	2352713	NICK'S FIRE GRILL STEAK & LEMONADE INC.	NICK'S FIRE GRILL STEAK & LEMONADE INC.	2699090		Risk 1 (High)	2900 W 63RD ST	CHICAGO	IL	60629	2019-12-04T00:00:00.000	License	Not Ready	
4	2352701	TAQUERIA BLUE LINE INC.		2703436	Restaurant	Risk 1 (High)	3401 W BELMONT AVE	CHICAGO	IL	60618	2019-12-04T00:00:00.000	License	Pass w/ Conditions	3. MANAGEMENT, FOOD EMPL
5	2352727	PORTAGE PARK DAY NURSERY	MOSAIC EARLY CHILDHOOD ACADEMY	2215815	Children's Services Facility	Risk 1 (High)	5332-5334 W ADDISON ST	CHICAGO	IL	60641	2019-12-04T00:00:00.000	Canvass	Pass	
6	2352738	AMARIT RESTAURANT	AMARIT RESTAURANT	1801618	Restaurant	Risk 1 (High)	600 S DEARBORN ST	CHICAGO	IL	60605	2019-12-04T00:00:00.000	Canvass Re-Inspection	Pass	

Code :

```
dim(df_Food)
```

Output :

```
[1] 196825     22
```

Code :

```
install.packages("dplyr")

library(dplyr)

glimpse(df_Food)
```

Output :

```
> glimpse(df_Food)
Rows: 196,825
Columns: 22
$ Inspection.ID             <int> 2352683, 2352734, 2352713, 2352701, 2352727, 2352738, 2352684, 2352702, 2352718, 2352696, 2352629, 2352613, 2352601, 2352658, 2352677,…
$ DBA.Name                  <fct> "CHIPOTLE MEXICAN GRILL", "CHILI'S T-I", "NICK'S FIRE GRILL STEAK & LEMONADE INC.", "TAQUERIA BLUE LINE INC.", "PORTAGE PARK DAY NURSE…
$ AKA.Name                  <fct> "CHIPOTLE MEXICAN GRILL", "CHILI'S (T1-B14)", "NICK'S FIRE GRILL STEAK & LEMONADE INC.", "", "MOSAIC EARLY CHILDHOOD ACADEMY", "AMARIT…
$ License..                 <int> 2670642, 34169, 2699090, 2703436, 2215815, 1801618, 2670643, 2698776, 23931, 2506383, 1170410, 2703724, 2689717, 2670643, 12144, 26898…
$ Facility.Type             <fct> "Restaurant", "Restaurant", "", "Restaurant", "Children's Services Facility", "Restaurant", "Restaurant", "Grocery Store", "School", "…
$ Risk                      <fct> Risk 1 (High), Risk 1 (High), Risk 1 (High), Risk 1 (High), Risk 1 (High), Risk 1 (High), Risk 1 (High), Risk 3 (Low), Risk 1 (High), …
$ Address                   <fct> 1025 W ADDISON ST , 11601 W TOUHY AVE , 2900 W 63RD ST , 3401 W BELMONT AVE , 5332-5334 W ADDISON ST , 600 S DEARBORN ST , 1025 W ADDI…
$ City                      <fct> CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO,…
$ State                     <fct> IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL…
$ Zip                       <int> 60613, 60666, 60629, 60618, 60641, 60605, 60613, 60657, 60659, 60641, 60622, 60614, 60618, 60613, 60630, 60638, 60605, 60654, 60661, 6…
$ Inspection.Date           <fct> 2019-12-04T00:00:00.000, 2019-12-04T00:00:00.000, 2019-12-04T00:00:00.000, 2019-12-04T00:00:00.000, 2019-12-04T00:00:00.000, 2019-12-0…
$ Inspection.Type           <fct> License Re-Inspection, Canvass, License, License, Canvass, Canvass Re-Inspection, License Re-Inspection, License, Canvass, Canvass, Ca…
$ Results                   <fct> Pass w/ Conditions, Pass, Not Ready, Pass w/ Conditions, Pass, Pass, Pass w/ Conditions, Pass w/ Conditions, Pass, Pass, Pass w/ Condi…
$ Violations                <fct> "3. MANAGEMENT, FOOD EMPLOYEE AND CONDITIONAL EMPLOYEE; KNOWLEDGE, RESPONSIBILITIES AND REPORTING - Comments: FOUND NO EMPLOYEE HEALTH…
$ Latitude                  <dbl> 41.94712, 42.00854, 41.77919, 41.93917, 41.94606, 41.87448, 41.94712, 41.94303, 41.98461, 41.95488, 41.90962, 41.91098, 41.94332, 41.9…
$ Longitude                 <dbl> -87.65538, -87.91443, -87.69589, -87.71255, -87.76072, -87.62936, -87.65538, -87.65300, -87.70057, -87.72751, -87.67759, -87.65121, -8…
$ Location                  <fct> "{'latitude': '-87.6553811622124', 'longitude': '41.94711774835235'}", "{'latitude': '-87.91442843927047', 'longitude': '42.0085364008…
$ Historical.Wards.2003.2015 <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
$ Zip.Codes                 <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
$ Community.Areas           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
$ Census.Tracts             <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
$ Wards                     <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
> |
```

Code :

colnames(df_Food)

Output :

```
> colnames(df_Food)
 [1] "Inspection.ID"             "DBA.Name"                  "AKA.Name"                  "License.."                 "Facility.Type"
 [6] "Risk"                      "Address"                   "City"                      "State"                     "Zip"
[11] "Inspection.Date"           "Inspection.Type"           "Results"                   "Violations"                "Latitude"
[16] "Longitude"                 "Location"                  "Historical.Wards.2003.2015" "Zip.Codes"                "Community.Areas"
[21] "Census.Tracts"             "Wards"
```

** Note : Clean the dataset by removing the columns which only consist no values

df_clean=df_Food[,-c(18:22)]

Q2.View the first 6 records.

Code :

View(head(df_clean,6))

Output :

Filter

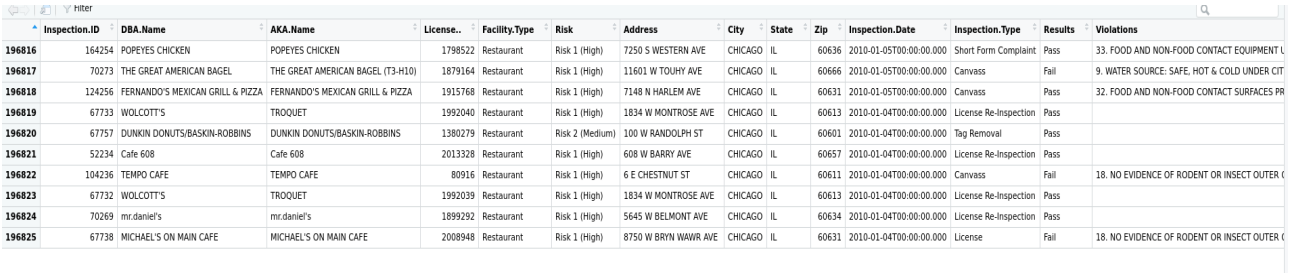
	Inspection.ID	DBA.Name	AKA.Name	License..	Facility.Type	Risk	Address	City	State	Zip	Inspection.Date	Inspection.Type	Results	Violations
1	2352683	CHIPOTLE MEXICAN GRILL	CHIPOTLE MEXICAN GRILL	2670642	Restaurant	Risk 1 (High)	1025 W ADDISON ST	CHICAGO	IL	60613	2019-12-04T00:00:00.000	License Re-Inspection	Pass w/ Conditions	3. MANAGEMENT, FOOD EMPL
2	2352734	CHILI'S T-I	CHILI'S (T1-B14)	34169	Restaurant	Risk 1 (High)	11601 W TOUHY AVE	CHICAGO	IL	60666	2019-12-04T00:00:00.000	Canvass	Pass	10. ADEQUATE HANDWASHIN
3	2352713	NICK'S FIRE GRILL STEAK & LEMONADE INC.	NICK'S FIRE GRILL STEAK & LEMONADE INC.	2699090		Risk 1 (High)	2900 W 63RD ST	CHICAGO	IL	60629	2019-12-04T00:00:00.000	License	Not Ready	
4	2352701	TAQUERIA BLUE LINE INC.		2703436	Restaurant	Risk 1 (High)	3401 W BELMONT AVE	CHICAGO	IL	60618	2019-12-04T00:00:00.000	License	Pass w/ Conditions	3. MANAGEMENT, FOOD EMPL
5	2352727	PORTAGE PARK DAY NURSERY	MOSAIC EARLY CHILDHOOD ACADEMY	2215815	Children's Services Facility	Risk 1 (High)	5332-5334 W ADDISON ST	CHICAGO	IL	60641	2019-12-04T00:00:00.000	Canvass	Pass	
6	2352738	AMARIT RESTAURANT	AMARIT RESTAURANT	1801618	Restaurant	Risk 1 (High)	600 S DEARBORN ST	CHICAGO	IL	60605	2019-12-04T00:00:00.000	Canvass Re-Inspection	Pass	

Q3.View the last 10 records

Code :

View(tail(df_clean,10))

Output :



	Inspection.ID	DBA.Name	AKA.Name	License..	Facility.Type	Risk	Address	City	State	Zip	Inspection.Date	Inspection.Type	Results	Violations
196816	164254	POPEYES CHICKEN	POPEYES CHICKEN	1798522	Restaurant	Risk 1 (High)	7250 S WESTERN AVE	CHICAGO	IL	60636	2010-01-05T00:00:00.000	Short Form Complaint	Pass	33. FOOD AND NON-FOOD CONTACT EQUIPMENT U
196817	70273	THE GREAT AMERICAN BAGEL	THE GREAT AMERICAN BAGEL (T3-H10)	1879164	Restaurant	Risk 1 (High)	11601 W TOUHY AVE	CHICAGO	IL	60666	2010-01-05T00:00:00.000	Canvass	Fail	9. WATER SOURCE: SAFE, HOT & COLD UNDER CIT
196818	124256	FERNANDO'S MEXICAN GRILL & PIZZA	FERNANDO'S MEXICAN GRILL & PIZZA	1915768	Restaurant	Risk 1 (High)	7148 N HARLEM AVE	CHICAGO	IL	60631	2010-01-05T00:00:00.000	Canvass	Pass	32. FOOD AND NON-FOOD CONTACT SURFACES PR
196819	67733	WOLCOTT'S	TROQUET	1992040	Restaurant	Risk 1 (High)	1834 W MONTROSE AVE	CHICAGO	IL	60613	2010-01-04T00:00:00.000	License Re-Inspection	Pass	
196820	67757	DUNKIN DONUTS/BASKIN-ROBBINS	DUNKIN DONUTS/BASKIN-ROBBINS	1380279	Restaurant	Risk 2 (Medium)	100 W RANDOLPH ST	CHICAGO	IL	60601	2010-01-04T00:00:00.000	Tag Removal	Pass	
196821	52234	Cafe 608	Cafe 608	2013328	Restaurant	Risk 1 (High)	608 W BARRY AVE	CHICAGO	IL	60657	2010-01-04T00:00:00.000	License Re-Inspection	Pass	
196822	104236	TEMPO CAFE	TEMPO CAFE	80916	Restaurant	Risk 1 (High)	6 E CHESTNUT ST	CHICAGO	IL	60611	2010-01-04T00:00:00.000	Canvass	Fail	18. NO EVIDENCE OF RODENT OR INSECT OUTER (
196823	67732	WOLCOTT'S	TROQUET	1992039	Restaurant	Risk 1 (High)	1834 W MONTROSE AVE	CHICAGO	IL	60613	2010-01-04T00:00:00.000	License Re-Inspection	Pass	
196824	70269	mr.daniel's	mr.daniel's	1899292	Restaurant	Risk 1 (High)	5645 W BELMONT AVE	CHICAGO	IL	60634	2010-01-04T00:00:00.000	License Re-Inspection	Pass	
196825	67738	MICHAEL'S ON MAIN CAFE	MICHAEL'S ON MAIN CAFE	2008948	Restaurant	Risk 1 (High)	8750 W BRYN WAWR AVE	CHICAGO	IL	60631	2010-01-04T00:00:00.000	License	Fail	18. NO EVIDENCE OF RODENT OR INSECT OUTER (

Q4.View the dimension of the dataset.

Code :

dim(df_clean)

Output :

[1] 196825 17

Q5.View the structure of the dataset

Code :

glimpse(df_clean)

Output :

```
> glimpse(df_clean)
Rows: 196,825
Columns: 17
$ Inspection.ID   <int> 2352683, 2352734, 2352713, 2352701, 2352727, 2352738, 2352684, 2352702, 2352718, 2352696, 2352629, 2352613, 2352601, 2352658, 2352677, 2352653, …
$ DBA.Name        <fct> "CHIPOTLE MEXICAN GRILL", "CHILI'S T-I", "NICK'S FIRE GRILL STEAK & LEMONADE INC.", "TAQUERIA BLUE LINE INC.", "PORTAGE PARK DAY NURSERY", "AMAR…
$ AKA.Name        <fct> "CHIPOTLE MEXICAN GRILL", "CHILI'S (T1-B14)", "NICK'S FIRE GRILL STEAK & LEMONADE INC.", "", "MOSAIC EARLY CHILDHOOD ACADEMY", "AMARIT RESTAURAN…
$ License..       <int> 2670642, 34169, 2699090, 2703436, 2215815, 1801618, 2670643, 2698776, 23931, 2506383, 1170410, 2703724, 2689717, 2670643, 12144, 2689813, 265326…
$ Facility.Type   <fct> "Restaurant", "Restaurant", "", "Restaurant", "Children's Services Facility", "Restaurant", "Restaurant", "Grocery Store", "School", "Children's…
$ Risk            <fct> Risk 1 (High), Risk 1 (High), Risk 1 (High), Risk 1 (High), Risk 1 (High), Risk 1 (High), Risk 1 (High), Risk 1 (High), Risk 3 (Low), Risk 1 (High), Risk 1 (Hi…
$ Address         <fct> 1025 W ADDISON ST , 11601 W TOUHY AVE , 2900 W 63RD ST , 3401 W BELMONT AVE , 5332-5334 W ADDISON ST , 600 S DEARBORN ST , 1025 W ADDISON ST , 3…
$ City            <fct> CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO, CHICAGO, …
$ State           <fct> IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, IL, …
$ Zip             <int> 60613, 60666, 60629, 60618, 60641, 60605, 60613, 60657, 60659, 60641, 60622, 60614, 60618, 60613, 60630, 60638, 60605, 60654, 60661, 60623, 6065…
$ Inspection.Date <fct> 2019-12-04T00:00:00.000, 2019-12-04T00:00:00.000, 2019-12-04T00:00:00.000, 2019-12-04T00:00:00.000, 2019-12-04T00:00:00.000, 2019-12-04T00:00:00…
$ Inspection.Type <fct> License Re-Inspection, Canvass, License, License, Canvass, Canvass Re-Inspection, License Re-Inspection, License, Canvass, Canvass, Canvass Re-I…
$ Results         <fct> Pass w/ Conditions, Pass, Not Ready, Pass w/ Conditions, Pass, Pass, Pass w/ Conditions, Pass w/ Conditions, Pass, Pass, Pass w/ Conditions, Not…
$ Violations      <fct> "3. MANAGEMENT, FOOD EMPLOYEE AND CONDITIONAL EMPLOYEE; KNOWLEDGE, RESPONSIBILITIES AND REPORTING - Comments: FOUND NO EMPLOYEE HEALTH POLICY AT…
$ Latitude        <dbl> 41.94712, 42.00854, 41.77919, 41.93917, 41.94606, 41.87448, 41.94712, 41.94303, 41.98461, 41.95488, 41.90962, 41.91098, 41.94332, 41.94712, 41.9…
$ Longitude       <dbl> -87.65538, -87.91443, -87.69589, -87.71255, -87.76072, -87.62936, -87.65538, -87.65300, -87.70057, -87.72751, -87.67759, -87.65121, -87.67943, -…
$ Location        <fct> "{'latitude': '-87.6553811622124', 'longitude': '41.94711774835235'}", "{'latitude': '-87.91442843927047', 'longitude': '42.008536400868735'}", …
```

Q6.View the names of the columns.

Code :

colnames(df_clean)

Output :

```
> colnames(df_clean)
 [1] "Inspection.ID"   "DBA.Name"        "AKA.Name"        "License.."       "Facility.Type"   "Risk"            "Address"         "City"            "State"
[10] "Zip"             "Inspection.Date" "Inspection.Type" "Results"         "Violations"      "Latitude"        "Longitude"       "Location"
>
```

Q7.View the 5 number summary for the numeric columns.

Code :

df_num = sapply(df_clean, is.numeric)

df_num = df_clean[df_num]

summary(df_num)

Output :

```
> df_num = sapply(df_clean, is.numeric)
> df_num = df_clean[df_num]
> summary(df_num)
 Inspection.ID       License..            Zip           Latitude       Longitude
 Min.   :  44247   Min.   :      0   Min.   :10014   Min.   :41.64   Min.   :-87.91
 1st Qu.:1150792   1st Qu.:1225118   1st Qu.:60614   1st Qu.:41.83   1st Qu.:-87.71
 Median :1493299   Median :1979973   Median :60625   Median :41.89   Median :-87.67
 Mean   :1449379   Mean   :1601134   Mean   :60629   Mean   :41.88   Mean   :-87.68
 3rd Qu.:2009233   3rd Qu.:2240843   3rd Qu.:60643   3rd Qu.:41.94   3rd Qu.:-87.64
 Max.   :2352738   Max.   :9999999   Max.   :60827   Max.   :42.02   Max.   :-87.53
                   NA's   :17        NA's   :50      NA's   :690     NA's   :690
```

Q8.Display the column with specific index number 3.

Code :

df_Food[,3]

Output :

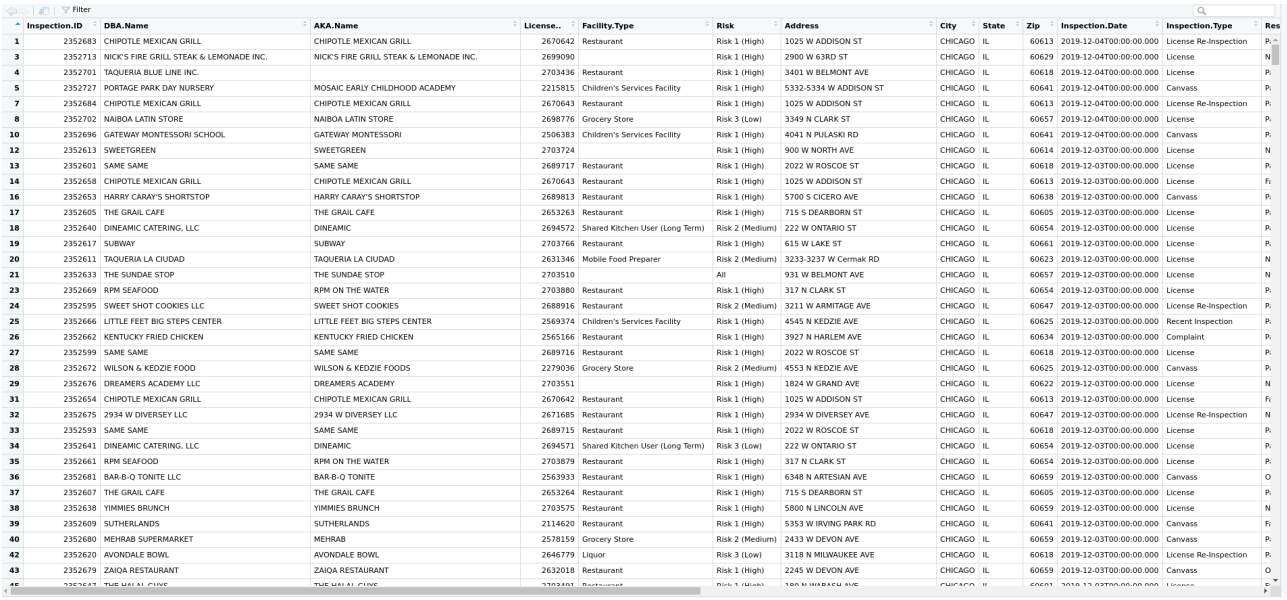
```
> df_Food[,3]
   [1] CHIPOTLE MEXICAN GRILL                      CHILI'S (T1-B14)                          NICK'S FIRE GRILL STEAK & LEMONADE INC.
   [4]                                             MOSAIC EARLY CHILDHOOD ACADEMY            AMARIT RESTAURANT
   [7] CHIPOTLE MEXICAN GRILL                      NAIBOA LATIN STORE                        JAMIESON PUBLIC SCHOOL
  [10] GATEWAY MONTESSORI                          BLUE LINE                                 SWEETGREEN
  [13] SAME SAME                                   CHIPOTLE MEXICAN GRILL                    BRIGADOON
  [16] HARRY CARAY'S SHORTSTOP                     THE GRAIL CAFE                            DINEAMIC
  [19] SUBWAY                                      TAQUERIA LA CIUDAD                        THE SUNDAE STOP
  [22] AMARIT RESTAURANT                           RPM ON THE WATER                          SWEET SHOT COOKIES
  [25] LITTLE FEET BIG STEPS CENTER                KENTUCKY FRIED CHICKEN                    SAME SAME
  [28] WILSON & KEDZIE FOODS                       DREAMERS ACADEMY                          CHICAGO SOCCER RESTAURANT
  [31] CHIPOTLE MEXICAN GRILL                      2934 W DIVERSEY LLC                       SAME SAME
  [34] DINEAMIC                                    RPM ON THE WATER                          BAR-B-Q TONITE
  [37] THE GRAIL CAFE                              YIMMIES BRUNCH                            SUTHERLANDS
  [40] MEHRAB                                      IMMACULATE CONCEPTION SCHOOL              AVONDALE BOWL
  [43] ZAIQA RESTAURANT                            CANTY ELEMENTARY SCHOOL                   THE HALAL GUYS
  [46] CITLALIN GALLERY/THEATER                    ELIZABETH  RESTAURANT                     TACOS & SALSAS
  [49] MCM PUB & LIQUORS                           PLUM MARKET/ INTELLIGENTSIA COFFEE        CLUB SALUD Y FELICIDAD
  [52] MASA AZUL                                   TC HOTDOGS                                Rio Grande Fruit Market
  [55] DANCEN                                      CHICAGO SMOKE                             RUNA JAPANESE
  [58] AROY THAI RESTAURANT                        DUNKIN DONUTS                             SEOUL DAN LAN II OR NOODLES SOUP
  [61] NICKY'S GRILL & YOGURT OASIS                LA MICHOACANA PREMIUM                     LOS COMALES #3
  [64] THE HALAL GUYS                              PALETERIA OSO POLAR                       TONY'S FINER FOODS
  [67] GEORGE HOT DOGS                             FLIPPIN FLAVORS                           HAMBURGER HEAVEN
  [70] WISE OWL DAYCARE                            HEARTY CAFE PANCAKE HOUSE                 CAFE RESTAURANT ART
  [73] CHICAGO MERCANTILE EXCHANGE INC.            REAL GOOD JUICE CO.                       VICKY'S CAFE
  [76] TITA'S DOGS                                 2 H CAFE + BAR                            EAT & DRINK
  [79] CHIKIS N GRILL                              BADOU SENEGALESE CUISINE                  CIBO FOODS,LLC
  [82] RABBITS BAR & GRILL                         ME DEE CAFE                               TAQUERIA MORELIA
  [85] OSAS AFRICAN RESTAURANT & CATERING SERVICES WISE OWL DRINKERY & COOKHOUSE             SHRADDHA WINE AND SPIRIT
  [88] SHRADDHA WINE AND SPIRIT                    THREE HARMONY RESTAURANT                  TAKITO STREET
  [91] HONEYBAKED HAM AND CAFE                     TESFA ETHIOPIAN CUISINE                   TAKITO STREET
  [94] ROREMOST LIQUORS                            AMIGOS FOOD MARKET                        MISS SAIGON
  [97] DUNKIN DONUTS                               SUBWAY                                    NEW BANH MI & CO.
```

Q9.To select the rows where license no between 20L to 30L.

Code :

View(df_Food[df_Food$License..>=2000000 & df_Food$License..<=3000000,])

Output :



	Inspection.ID	DBA.Name	AKA.Name	License..	Facility.Type	Risk	Address	City	State	Zip	Inspection.Date	Inspection.Type	Res
1	2352683	CHIPOTLE MEXICAN GRILL	CHIPOTLE MEXICAN GRILL	2670642	Restaurant	Risk 1 (High)	1025 W ADDISON ST	CHICAGO	IL	60613	2019-12-04T00:00:00.000	License Re-Inspection	P
3	2352713	NICK'S FIRE GRILL STEAK & LEMONADE INC.	NICK'S FIRE GRILL STEAK & LEMONADE INC.	2699090		Risk 1 (High)	2900 W 63RD ST	CHICAGO	IL	60629	2019-12-04T00:00:00.000	License	N
4	2352701	TAQUERIA BLUE LINE INC.		2703436	Restaurant	Risk 1 (High)	3401 W BELMONT AVE	CHICAGO	IL	60618	2019-12-04T00:00:00.000	License	P
5	2352727	PORTAGE PARK DAY NURSERY	MOSAIC EARLY CHILDHOOD ACADEMY	2215815	Children's Services Facility	Risk 1 (High)	5332-5334 W ADDISON ST	CHICAGO	IL	60641	2019-12-04T00:00:00.000	Canvass	P
7	2352684	CHIPOTLE MEXICAN GRILL	CHIPOTLE MEXICAN GRILL	2670643	Restaurant	Risk 1 (High)	1025 W ADDISON ST	CHICAGO	IL	60613	2019-12-04T00:00:00.000	License Re-Inspection	P
8	2352702	NAIBOA LATIN STORE	NAIBOA LATIN STORE	2698776	Grocery Store	Risk 3 (Low)	3349 N CLARK ST	CHICAGO	IL	60657	2019-12-04T00:00:00.000	License	P
10	2352696	GATEWAY MONTESSORI SCHOOL	GATEWAY MONTESSORI	2506383	Children's Services Facility	Risk 1 (High)	4041 N PULASKI RD	CHICAGO	IL	60641	2019-12-04T00:00:00.000	Canvass	P
12	2352613	SWEETGREEN	SWEETGREEN	2703724		Risk 1 (High)	900 W NORTH AVE	CHICAGO	IL	60614	2019-12-03T00:00:00.000	License	N
13	2352601	SAME SAME	SAME SAME	2689717	Restaurant	Risk 1 (High)	2022 W ROSCOE ST	CHICAGO	IL	60618	2019-12-03T00:00:00.000	License	P
14	2352658	CHIPOTLE MEXICAN GRILL	CHIPOTLE MEXICAN GRILL	2670643	Restaurant	Risk 1 (High)	1025 W ADDISON ST	CHICAGO	IL	60613	2019-12-03T00:00:00.000	License	Fa
16	2352653	HARRY CARAY'S SHORTSTOP	HARRY CARAY'S SHORTSTOP	2689813	Restaurant	Risk 1 (High)	5700 S CICERO AVE	CHICAGO	IL	60638	2019-12-03T00:00:00.000	Canvass	P
17	2352605	THE GRAIL CAFE	THE GRAIL CAFE	2653263	Restaurant	Risk 1 (High)	715 S DEARBORN ST	CHICAGO	IL	60605	2019-12-03T00:00:00.000	License	P
18	2352640	DINEAMIC CATERING, LLC	DINEAMIC	2694572	Shared Kitchen User (Long Term)	Risk 2 (Medium)	222 W ONTARIO ST	CHICAGO	IL	60654	2019-12-03T00:00:00.000	License	P
19	2352617	SUBWAY	SUBWAY	2703766	Restaurant	Risk 1 (High)	615 W LAKE ST	CHICAGO	IL	60661	2019-12-03T00:00:00.000	License	P
20	2352611	TAQUERIA LA CIUDAD	TAQUERIA LA CIUDAD	2631346	Mobile Food Preparer	Risk 2 (Medium)	3233-3237 W Cermak RD	CHICAGO	IL	60623	2019-12-03T00:00:00.000	License	N
21	2352633	THE SUNDAE STOP	THE SUNDAE STOP	2703510		All	931 W BELMONT AVE	CHICAGO	IL	60657	2019-12-03T00:00:00.000	License	N
23	2352669	RPM SEAFOOD	RPM ON THE WATER	2703880	Restaurant	Risk 1 (High)	317 N CLARK ST	CHICAGO	IL	60654	2019-12-03T00:00:00.000	License	P
24	2352595	SWEET SHOT COOKIES LLC	SWEET SHOT COOKIES	2688916	Restaurant	Risk 2 (Medium)	3211 W ARMITAGE AVE	CHICAGO	IL	60647	2019-12-03T00:00:00.000	License Re-Inspection	P
25	2352666	LITTLE FEET BIG STEPS CENTER	LITTLE FEET BIG STEPS CENTER	2569374	Children's Services Facility	Risk 1 (High)	4545 N KEDZIE AVE	CHICAGO	IL	60625	2019-12-03T00:00:00.000	Recent Inspection	P
26	2352662	KENTUCKY FRIED CHICKEN	KENTUCKY FRIED CHICKEN	2565166	Restaurant	Risk 1 (High)	3927 N HARLEM AVE	CHICAGO	IL	60634	2019-12-03T00:00:00.000	Complaint	P
27	2352599	SAME SAME	SAME SAME	2689716	Restaurant	Risk 1 (High)	2022 W ROSCOE ST	CHICAGO	IL	60618	2019-12-03T00:00:00.000	License	P
28	2352672	WILSON & KEDZIE FOOD	WILSON & KEDZIE FOODS	2279036	Grocery Store	Risk 2 (Medium)	4553 N KEDZIE AVE	CHICAGO	IL	60625	2019-12-03T00:00:00.000	Canvass	P
29	2352676	DREAMERS ACADEMY LLC	DREAMERS ACADEMY	2703551		Risk 1 (High)	1824 W GRAND AVE	CHICAGO	IL	60622	2019-12-03T00:00:00.000	License	N
31	2352654	CHIPOTLE MEXICAN GRILL	CHIPOTLE MEXICAN GRILL	2670642	Restaurant	Risk 1 (High)	1025 W ADDISON ST	CHICAGO	IL	60613	2019-12-03T00:00:00.000	License	Fa
32	2352675	2934 W DIVERSEY LLC	2934 W DIVERSEY LLC	2671685	Restaurant	Risk 1 (High)	2934 W DIVERSEY AVE	CHICAGO	IL	60647	2019-12-03T00:00:00.000	License Re-Inspection	N
33	2352593	SAME SAME	SAME SAME	2689715	Restaurant	Risk 1 (High)	2022 W ROSCOE ST	CHICAGO	IL	60618	2019-12-03T00:00:00.000	License	P
34	2352641	DINEAMIC CATERING, LLC	DINEAMIC	2694571	Shared Kitchen User (Long Term)	Risk 3 (Low)	222 W ONTARIO ST	CHICAGO	IL	60654	2019-12-03T00:00:00.000	License	P
35	2352661	RPM SEAFOOD	RPM ON THE WATER	2703879	Restaurant	Risk 1 (High)	317 N CLARK ST	CHICAGO	IL	60654	2019-12-03T00:00:00.000	License	P
36	2352681	BAR-B-Q TONITE LLC	BAR-B-Q TONITE	2563933	Restaurant	Risk 1 (High)	6348 N ARTESIAN AVE	CHICAGO	IL	60659	2019-12-03T00:00:00.000	Canvass	O
37	2352607	THE GRAIL CAFE	THE GRAIL CAFE	2653264	Restaurant	Risk 1 (High)	715 S DEARBORN ST	CHICAGO	IL	60605	2019-12-03T00:00:00.000	License	P
38	2352638	YIMMIES BRUNCH	YIMMIES BRUNCH	2703575	Restaurant	Risk 1 (High)	5800 N LINCOLN AVE	CHICAGO	IL	60659	2019-12-03T00:00:00.000	License	N
39	2352609	SUTHERLANDS	SUTHERLANDS	2114620	Restaurant	Risk 1 (High)	5353 W IRVING PARK RD	CHICAGO	IL	60641	2019-12-03T00:00:00.000	Canvass	Fa
40	2352680	MEHRAB SUPERMARKET	MEHRAB	2578159	Grocery Store	Risk 2 (Medium)	2433 W DEVON AVE	CHICAGO	IL	60659	2019-12-03T00:00:00.000	Canvass	P
42	2352620	AVONDALE BOWL	AVONDALE BOWL	2646779	Liquor	Risk 3 (Low)	3118 N MILWAUKEE AVE	CHICAGO	IL	60618	2019-12-03T00:00:00.000	License Re-Inspection	P
43	2352679	ZAIQA RESTAURANT	ZAIQA RESTAURANT	2632018	Restaurant	Risk 1 (High)	2245 W DEVON AVE	CHICAGO	IL	60659	2019-12-03T00:00:00.000	Canvass	O

Q10.To select the records where DBA.Name =“Blue Line”

Code :

View(df_Food[df_Food$DBA.Name=="BLUE LINE",])

Output :

Filter

	Inspection.ID	DBA.Name	AKA.Name	License..	Facility.Type	Risk	Address	City	State	Zip	Inspection.Date	Inspection.Type	Results	Violations	Latitude	Longitude	Location
11	2352629	BLUE LINE	BLUE LINE	1170410	Restaurant	Risk 1 (High)	1548 N DAMEN AVE	CHICAGO	IL	60622	2019-12-03T00:00:00.000	Canvass Re-Inspection	Pass w/ Conditions	25. CONSUMER ADVISORY PROVIDED FOR RAW/UND...	41.90962	-87.67759	{'latitude
121	2352363	BLUE LINE	BLUE LINE	1170410	Restaurant	Risk 1 (High)	1548 N DAMEN AVE	CHICAGO	IL	60622	2019-11-26T00:00:00.000	Canvass	Fail	23. PROPER DATE MARKING AND DISPOSITION - Co...	41.90962	-87.67759	{'latitude
346	2346014	BLUE LINE	BLUE LINE	1170410	Restaurant	Risk 1 (High)	1548 N DAMEN AVE	CHICAGO	IL	60622	2019-11-20T00:00:00.000	Canvass	No Entry		41.90962	-87.67759	{'latitude
618	2345546	BLUE LINE	BLUE LINE	1170410	Restaurant	Risk 1 (High)	1548 N DAMEN AVE	CHICAGO	IL	60622	2019-11-13T00:00:00.000	Canvass	No Entry		41.90962	-87.67759	{'latitude
20831	2222320	BLUE LINE	BLUE LINE	1170410	Restaurant	Risk 1 (High)	1548 N DAMEN AVE	CHICAGO	IL	60622	2018-09-13T00:00:00.000	Canvass	Pass w/ Conditions	3. MANAGEMENT, FOOD EMPLOYEE AND CONDITION...	41.90962	-87.67759	{'latitude
40127	2081467	BLUE LINE	BLUE LINE	1170410	Restaurant	Risk 1 (High)	1548 N DAMEN AVE	CHICAGO	IL	60622	2017-09-01T00:00:00.000	Canvass	Pass w/ Conditions	3. POTENTIALLY HAZARDOUS FOOD MEETS TEMPER...	41.90962	-87.67759	{'latitude
41293	2078375	BLUE LINE	BLUE LINE	1170410	Restaurant	Risk 1 (High)	1548 N DAMEN AVE	CHICAGO	IL	60622	2017-08-14T00:00:00.000	Canvass	No Entry		41.90962	-87.67759	{'latitude
71346	1763379	BLUE LINE	BLUE LINE	1170410	Restaurant	Risk 1 (High)	1548 N DAMEN AVE	CHICAGO	IL	60622	2016-04-21T00:00:00.000	Canvass	Pass	32. FOOD AND NON-FOOD CONTACT SURFACES PRO...	41.90962	-87.67759	{'latitude
93042	1453901	BLUE LINE	BLUE LINE	1170410	Restaurant	Risk 1 (High)	1548 N DAMEN AVE	CHICAGO	IL	60622	2015-04-20T00:00:00.000	Canvass	No Entry		41.90962	-87.67759	{'latitude
110011	1490295	BLUE LINE	BLUE LINE	1170410	Restaurant	Risk 1 (High)	1548 N DAMEN AVE	CHICAGO	IL	60622	2014-07-02T00:00:00.000	Canvass	Pass	38. VENTILATION: ROOMS AND EQUIPMENT VENTED...	41.90962	-87.67759	{'latitude
125689	1361374	BLUE LINE	BLUE LINE	1170410	Restaurant	Risk 1 (High)	1548 N DAMEN AVE	CHICAGO	IL	60622	2013-09-27T00:00:00.000	Canvass	Pass	31. CLEAN MULTI-USE UTENSILS AND SINGLE SERVI...	41.90962	-87.67759	{'latitude
163724	608298	BLUE LINE	BLUE LINE	1170410	Restaurant	Risk 1 (High)	1548 N DAMEN AVE	CHICAGO	IL	60622	2011-10-18T00:00:00.000	Canvass	Pass	32. FOOD AND NON-FOOD CONTACT SURFACES PRO...	41.90962	-87.67759	{'latitude

Q11.To display only Inspection.id and Inspection.Type.

Code :

```
View(df_Food[,c("Inspection.ID","Inspection.Type")])
```

Output :

Filter

	Inspection.ID	Inspection.Type
1	2352683	License Re-Inspection
2	2352734	Canvass
3	2352713	License
4	2352701	License
5	2352727	Canvass
6	2352738	Canvass Re-Inspection
7	2352684	License Re-Inspection
8	2352702	License
9	2352718	Canvass
10	2352696	Canvass
11	2352629	Canvass Re-Inspection
12	2352613	License
13	2352601	License
14	2352658	License
15	2352677	Complaint Re-Inspection
16	2352653	Canvass
17	2352605	License
18	2352640	License
19	2352617	License
20	2352611	License
21	2352633	License
22	2352649	Canvass
23	2352669	License
24	2352595	License Re-Inspection
25	2352666	Recent Inspection
26	2352662	Complaint
27	2352599	License
28	2352672	Canvass
29	2352676	License
30	2352646	Canvass Re-Inspection
31	2352654	License
32	2352675	License Re-Inspection
33	2352593	License
34	2352641	License
35	2352661	License
36	2352681	Canvass

Showing 1 to 37 of 196,825 entries, 2 total columns

Q12.Select the names of DBA where risk is high.

Code :

df_Food[df_Food$Risk=="Risk 1 (High)","DBA.Name"]

Output :

```
> df_Food[df_Food$Risk=="Risk 1 (High)","DBA.Name"]
   [1] CHIPOTLE MEXICAN GRILL                 CHILI'S T-I                                NICK'S FIRE GRILL STEAK & LEMONADE INC.
   [4] TAQUERIA BLUE LINE INC.                PORTAGE PARK DAY NURSERY                   AMARIT RESTAURANT
   [7] CHIPOTLE MEXICAN GRILL                 JAMIESON                                   GATEWAY MONTESSORI SCHOOL
  [10] BLUE LINE                              SWEETGREEN                                 SAME SAME
  [13] CHIPOTLE MEXICAN GRILL                 HARRY CARAY'S SHORTSTOP                    THE GRAIL CAFE
  [16] SUBWAY                                 AMARIT RESTAURANT                          RPM SEAFOOD
  [19] LITTLE FEET BIG STEPS CENTER           KENTUCKY FRIED CHICKEN                     SAME SAME
  [22] DREAMERS ACADEMY LLC                   CHICAGO SOCCER RESTAURANT                  CHIPOTLE MEXICAN GRILL
  [25] 2934 W DIVERSEY LLC                    SAME SAME                                  RPM SEAFOOD
  [28] BAR-B-Q TONITE LLC                     THE GRAIL CAFE                             YIMMIES BRUNCH
  [31] SUTHERLANDS                            IMMACULATE CONCEPTION SCHOOL               ZAIQA RESTAURANT
  [34] ARTHUR CANTY ELEMENTARY                THE HALAL GUYS                             CITLALIN GALLERY/THEATER
  [37] ELIZABETH  RESTAURANT                  MCM PUB & LIQUORS                          PLUM MARKET
  [40] MASA AZUL                              Rio Grande Fruit Market, Inc.              DANCEN GRILL RESTAURANT, INC.
  [43] CHICAGO SMOKE LTD                      RUNA JAPANESE                              AROY THAI RESTAURANT
  [46] YA KWANG                               LOS COMALES #3                             THE HALAL GUYS
  [49] PALETERIA OSO POLAR, INC               TONY'S FINER FOODS                         GEORGE HOT DOGS
  [52] FLIPPIN FLAVORS                        HAMBURGER HEAVEN                           WISE OWL DAYCARE LLC
  [55] HEARTY CAFE PANCAKE HOUSE              CAFE RESTAURANT ART                        REAL GOOD JUICE CO.
  [58] VICKY'S CAFE LLC                       LIBERO CAFE                                EAT & DRINK, INC
  [61] CHIKIS N GRILL                         BADOU SENEGALESE CUISINE                   ME DEE INC.
  [64] TAQUERIA MORELIA                       OSAS AFRICAN RESTAURANT & CATERING SERVICES WISE OWL DRINKERY & COOKHOUSE
  [67] THREE HARMONY RESTAURANT               HONEYBAKED HAM AND CAFE                    TESFA ETHIOPIAN CUISINE
  [70] TAKITO STREET                          MISS SAIGON                                SUBWAY / TCBY
  [73] NEW BANH MI & CO.                      JUANITA'S RESTAURANT #2                    CITY PROVISIONS CATERING AND E
  [76] GOLDEN CHEF CHINESE KITCHEN            CAROLINE BBQ, INC                          KYOTEN
  [79] PINK MONKEY / NEW YORK STRIP           ZEE'S FOOD MARKET INC.                     DAVE'S FOOD MART
  [82] Hands On Inc                           NEW HOPE HEAD START                        NIA
  [85] M&M RESTAURANT GROUP                   HOOTS ON DIVERSEY                          BLUE LINE
  [88] O'cha Thai Cuisine                     LIBERO CAFE                                HANDCUT FOODS
  [91] ETTA                                   BLVD CHICAGO                               CREATIVE SCHOLARS PRESCHOOL
  [94] MARISQUERIA LOS CABOS                  KFC                                        PINKS CHILD CARE ACADEMY  II
  [97] TAKITO STREET                          GO! GROCER #4                              PINOCCHIO CHILD CARE
```

Q13.Select the first 2 columns where result of inspection is fail.

Code :

View(df_Food[df_Food$Results=="Fail",c(1,2)])

Output :

Filter

	Inspection.ID	DBA.Name
14	2352658	CHIPOTLE MEXICAN GRILL
22	2352649	AMARIT RESTAURANT
31	2352654	CHIPOTLE MEXICAN GRILL
39	2352609	SUTHERLANDS
45	2352647	THE HALAL GUYS
46	2352583	CITLALIN GALLERY/THEATER
54	2352525	Rio Grande Fruit Market, Inc.
61	2352567	NICKY'S GRILL & YOGURT OASIS
62	2352575	LA RICA MICHOACANA NEVERIA, LLC
68	2352545	FLIPPIN FLAVORS
86	2352471	WISE OWL DRINKERY & COOKHOUSE
94	2352466	FOREMOST LIQUORS
98	2352409	SUBWAY / TCBY
121	2352363	BLUE LINE
122	2352332	NEMA CHICAGO
124	2352330	LIBERO CAFE
132	2352328	SALGADO'S BAKERY
139	2352327	NEMA CHICAGO
140	2352335	TAKITO STREET
141	2352345	GO! GROCER #4
144	2352339	CHICAGO MERCANTILE EXCHANGE INC.
148	2352340	TAKITO STREET
150	2352364	SWEET SHOT COOKIES LLC
154	2352399	EL CAPITAN RESTAURANT
162	2352268	Rio Grande Fruit Market, Inc.
167	2352299	MR. SUBMARINE
168	2352269	LOCAL MARKET
169	2352261	PINK MONKEY / NEW YORK STRIP
171	2352282	MJ ONE STOP SHOP
181	2352289	MISS SAIGON
182	2352301	DUCK DUCK GOAT
183	2352254	DUNKIN DONUTS
189	2352266	LOCAL MARKET
192	2352296	STARBUCKS COFFEE #13522
197	2352300	TASTY CHINA
199	2352275	TAQUERIA MI BARRIO ES TEPITO

Showing 1 to 37 of 38,087 entries, 2 total columns

Q14.Check the total number of rows in the data frame.

Code :

```
nrow(df_Food)
```

Output :

```
> nrow(df_Food)

[1] 196825
```

Q15.To find the number of missing values column-wise.

Code :

```
sapply(df_Food, function(x)(sum(is.na(x))))
```

Output :

```
> sapply(df_Food, function(x)(sum(is.na(x))))
            Inspection.ID                 DBA.Name                 AKA.Name                License..            Facility.Type                     Risk 
                        0                        0                        0                       17                        0                        0 
                  Address                     City                    State                      Zip          Inspection.Date           Inspection.Type 
                        0                        0                        0                       50                        0                        0 
                  Results               Violations                 Latitude                Longitude         Location Historical.Wards.2003.2015 
                        0                        0                      690                      690                        0                   196825 
                Zip.Codes          Community.Areas            Census.Tracts                    Wards 
                   196825                   196825                   196825                   196825 
```

Q16.Delete the columns which have only null values.

Code :

ncol(df_Food)

null.col=sapply(df_Food, function(x)(sum(is.na(x))==nrow(df_Food)))

null.col

df_clean=df_Food[,-c(18:22)]

sapply(df_clean, function(x)(sum(is.na(x))==nrow(df_Food)))

ncol(df_clean)

colnames(df_clean)

Output :

```
> ncol(df_Food)
[1] 22
> null.col=sapply(df_Food, function(x)(sum(is.na(x))==nrow(df_Food)))
> null.col
            Inspection.ID                 DBA.Name                 AKA.Name                License..            Facility.Type                     Risk 
                    FALSE                    FALSE                    FALSE                    FALSE                    FALSE                    FALSE 
                  Address                     City                    State                      Zip          Inspection.Date           Inspection.Type 
                    FALSE                    FALSE                    FALSE                    FALSE                    FALSE                    FALSE 
                  Results               Violations                 Latitude                Longitude         Location Historical.Wards.2003.2015 
                    FALSE                    FALSE                    FALSE                    FALSE                    FALSE                     TRUE 
                Zip.Codes          Community.Areas            Census.Tracts                    Wards 
                     TRUE                     TRUE                     TRUE                     TRUE 
> df_clean=df_Food[,-c(18:22)]
> sapply(df_clean, function(x)(sum(is.na(x))==nrow(df_Food)))
  Inspection.ID        DBA.Name        AKA.Name       License..   Facility.Type            Risk         Address            City           State             Zip 
          FALSE           FALSE           FALSE           FALSE           FALSE           FALSE           FALSE           FALSE           FALSE           FALSE 
Inspection.Date Inspection.Type         Results      Violations        Latitude       Longitude        Location 
          FALSE           FALSE           FALSE           FALSE           FALSE           FALSE           FALSE 
> ncol(df_clean)
[1] 17
> colnames(df_clean)
 [1] "Inspection.ID"   "DBA.Name"        "AKA.Name"        "License.."       "Facility.Type"   "Risk"            "Address"         "City"            "State"          
[10] "Zip"             "Inspection.Date" "Inspection.Type" "Results"         "Violations"      "Latitude"        "Longitude"       "Location"
```

Q17.Draw a histogram of Latitude and longitude columns.

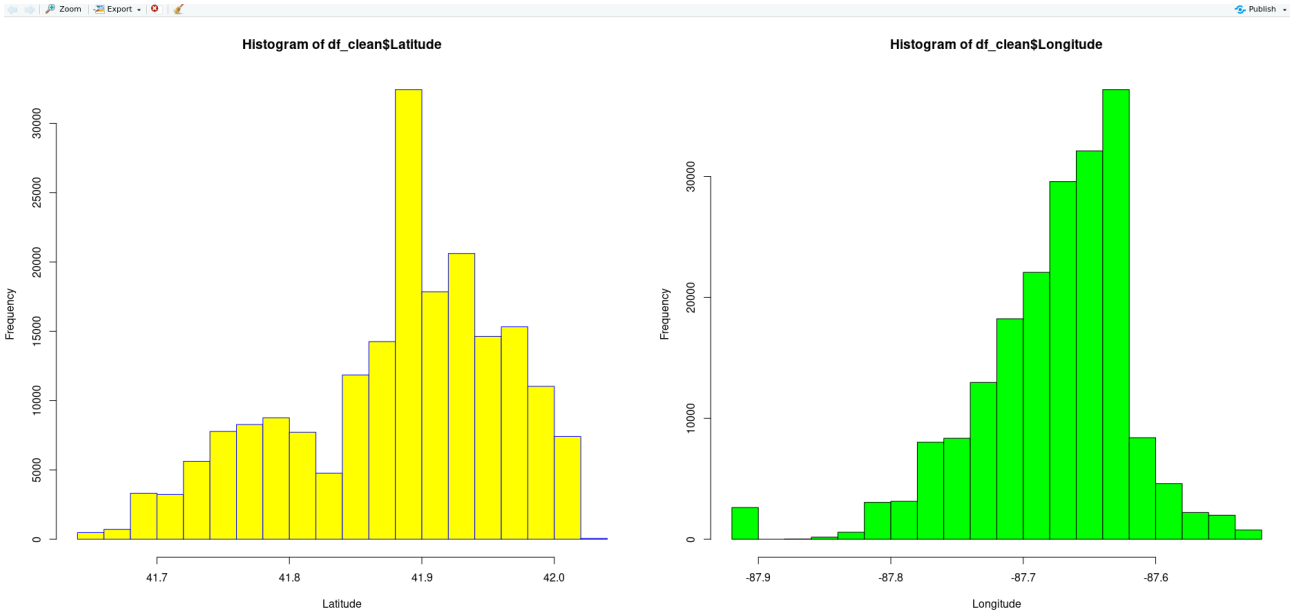
Code :

par(mfrow=c(1,2))

hist(df_clean$Latitude,xlab = "Latitude",col = "yellow",border = "blue")

hist(df_clean$Longitude,xlab = "Longitude",col = "green",border = "black")

Output :



Q18.Replace the missing values in Latitude and Longitude with mean.

Code :

df_clean$Latitude[is.na(df_clean$Latitude)] = mean(df_clean$Latitude,na.rm = TRUE)

df_clean$Longitude[is.na(df_clean$Longitude)] = mean(df_clean$Longitude,na.rm = TRUE)

sum(is.na(df_clean$Latitude))

sum(is.na(df_clean$Longitude))

Output :

> sum(is.na(df_clean$Latitude))

[1] 0

> sum(is.na(df_clean$Longitude))

[1] 0

Q19. Print the total number of duplicated values column-wise.

Code :

```
sapply(df_clean, function(x)(sum(duplicated(x))))
```

Output :

```
> sapply(df_clean, function(x)(sum(duplicated(x))))
  Inspection.ID        DBA.Name        AKA.Name        License..   Facility.Type            Risk         Address            City           State             Zip 
            149          169204          170478          159360          196333          196820          178744          196753          196820          196712 
Inspection.Date Inspection.Type         Results      Violations        Latitude       Longitude        Location 
         194305          196716          196818           53211          179989          179989          179981 
```

Q20.Display the count of PASS results grouped by type of risk.

Code :

```
library(dplyr)

df_clean %>% filter(Results=="Pass") %>% group_by(Risk) %>% summarise("Count"=n())
```

Output :

```
# A tibble: 4 × 2
  Risk              Count
  <fct>             <int>
1 ""                   12
2 "Risk 1 (High)"    78054
3 "Risk 2 (Medium)" 20595
4 "Risk 3 (Low)"     7405
```

Q21.Separate the inspection date into 3 (day ,month and year) columns.

Code :

```
df_clean$Inspection.Date = as.Date(df_clean$Inspection.Date)

df_clean$Day = as.numeric(format(df_clean$Inspection.Date,format = "%d"))

df_clean$month = as.numeric(format(df_clean$Inspection.Date,format = "%m"))

df_clean$year = as.numeric(format(df_clean$Inspection.Date,format = "%Y"))

df_clean[c(1:20),c("Inspection.Date","Day","month","year")]
```

Output :

```
> df_clean$Inspection.Date = as.Date(df_clean$Inspection.Date)
> df_clean$Day = as.numeric(format(df_clean$Inspection.Date,format = "%d"))
> df_clean$month = as.numeric(format(df_clean$Inspection.Date,format = "%m"))
> df_clean$year = as.numeric(format(df_clean$Inspection.Date,format = "%Y"))
> df_clean[c(1:20),c("Inspection.Date","Day","month","year")]
   Inspection.Date Day month year
1       2019-12-04   4    12 2019
2       2019-12-04   4    12 2019
3       2019-12-04   4    12 2019
4       2019-12-04   4    12 2019
5       2019-12-04   4    12 2019
6       2019-12-04   4    12 2019
7       2019-12-04   4    12 2019
8       2019-12-04   4    12 2019
9       2019-12-04   4    12 2019
10      2019-12-04   4    12 2019
11      2019-12-03   3    12 2019
12      2019-12-03   3    12 2019
13      2019-12-03   3    12 2019
14      2019-12-03   3    12 2019
15      2019-12-03   3    12 2019
16      2019-12-03   3    12 2019
17      2019-12-03   3    12 2019
18      2019-12-03   3    12 2019
19      2019-12-03   3    12 2019
20      2019-12-03   3    12 2019
```

Name : Sumon Singh

Assignment : Practical 2 Exploratory data analysis and visualisation using Python

Course : M.sc Computer Science With Specialization in Data Science

Paper : PSDS103

Analysing grade_score.csv

In [41]:
```
import pandas as pd
```

1.Import the dataset grade_score.csv

In [42]:
```
df_grade=pd.read_csv('/home/sumon/Downloads/grade_score.csv')
```

Checking the number of rows and columns of dataset

In [43]:
```
df_grade.shape
```

Out[43]:
```
(104, 7)
```

2.Print the data types of each columns

In [44]:
```
df_grade.dtypes
```

Out[44]:
```
Exam1            int64
Exam2            int64
Exam3            int64
Exam4          float64
Final_score    float64
Grade           object
Pass/fail       object
dtype: object
```

3.To print the number of the missing values in the dataset columnwise

In [45]:
```
df_grade.isna().sum()
```

Out[45]:
```
Exam1          0
Exam2          0
Exam3          0
Exam4          0
Final_score    0
```

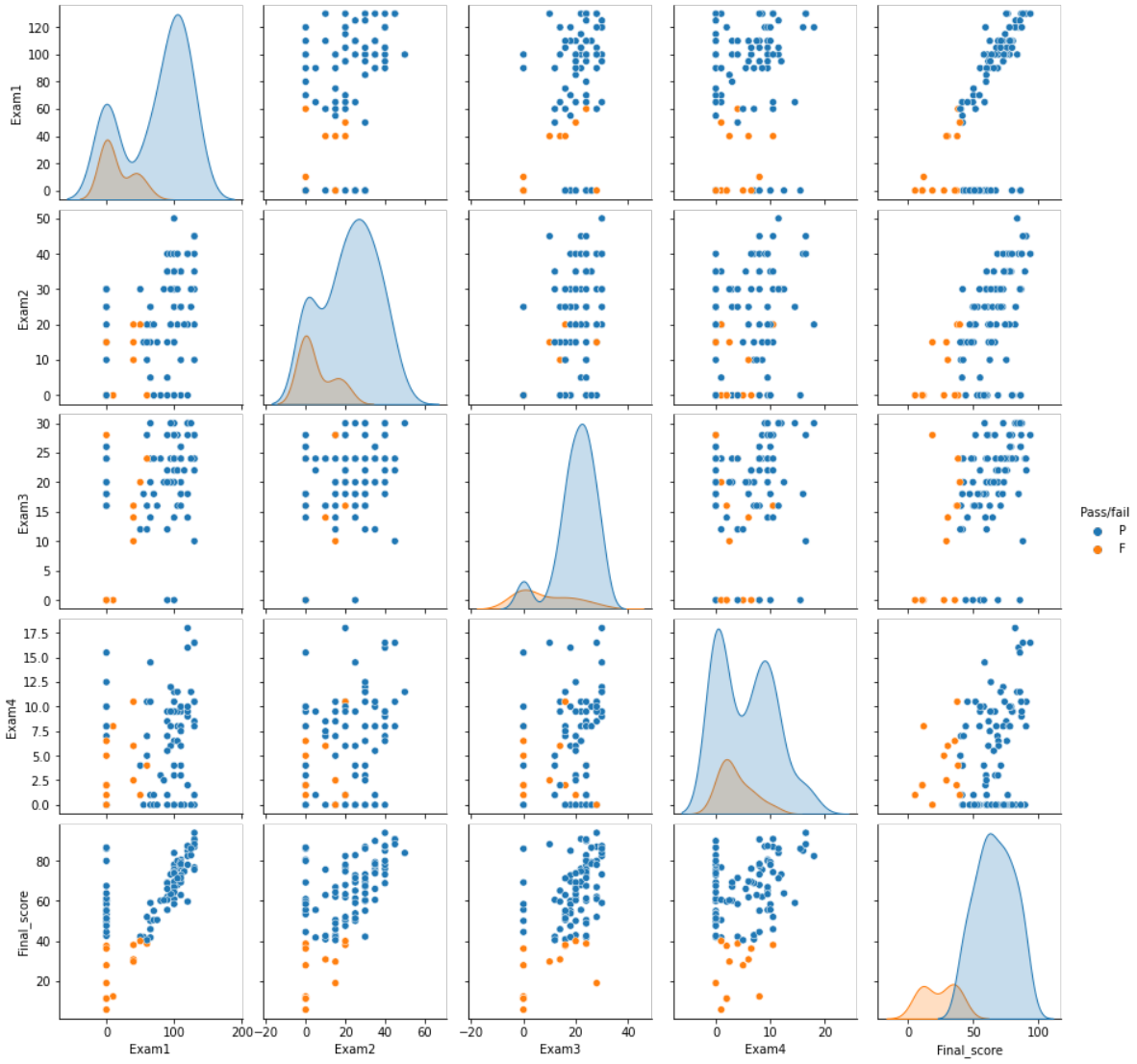
```
Grade          0
Pass/fail      0
```

4.Analyse the linear relatinship between features using pairplot

In [46]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [47]:
```
sns.pairplot(df_grade,hue="Pass/fail")
```

Out[47]:
```
<seaborn.axisgrid.PairGrid at 0x7fd68b320ee0>
```



5.Print the highest marks obtained in Exam1

In [48]:
```
df_grade["Exam1"].max()
```

Out[48]:
```
130
```

6.Print the total students grade-wise

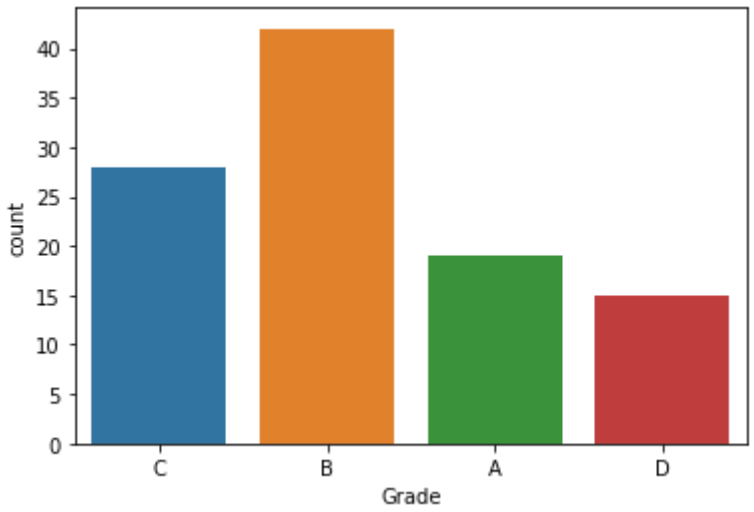
In [50]:
```
df_grade.groupby("Grade")["Grade"].count()
```

Out[50]:
```
Grade
A    19
B    42
C    28
D    15
Name: Grade, dtype: int64
```

Visualise the number of students gradewise

In [51]:
```
sns.countplot(x="Grade",data=df_grade)
```

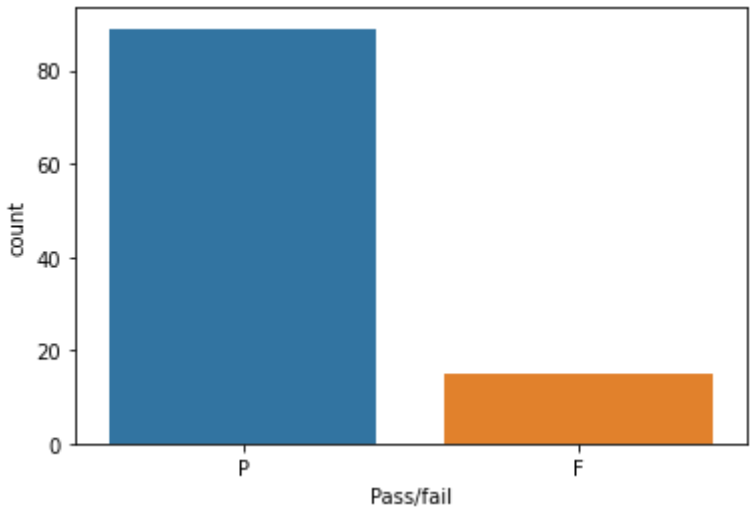
Out[51]:
```
<AxesSubplot:xlabel='Grade', ylabel='count'>
```



7.Visualise the number of students passed or failed

In [52]:
```
sns.countplot(x="Pass/fail",data=df_grade)
```

Out[52]:
```
<AxesSubplot:xlabel='Pass/fail', ylabel='count'>
```



8.Print the correlation between the numeric variable

In [53]:
```
df_num = df_grade.iloc[:,0:5]
df_num.corr()
```

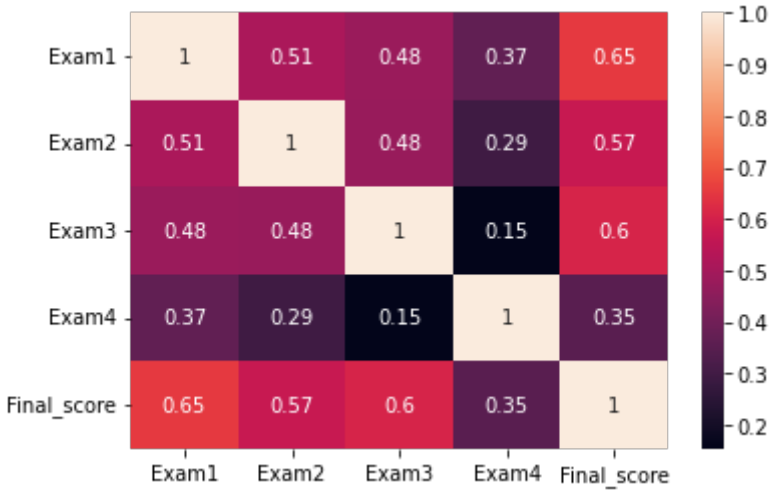
Out[53]:

	Exam1	Exam2	Exam3	Exam4	Final_score
Exam1	1.000000	0.511808	0.478851	0.367069	0.651011
Exam2	0.511808	1.000000	0.476269	0.293080	0.568483
Exam3	0.478851	0.476269	1.000000	0.153772	0.604635
Exam4	0.367069	0.293080	0.153772	1.000000	0.351176
Final_score	0.651011	0.568483	0.604635	0.351176	1.000000

9.Print the heatmap of the grade_score dataset

In [54]:
```
sns.heatmap(df_num.corr(),annot=True)
```

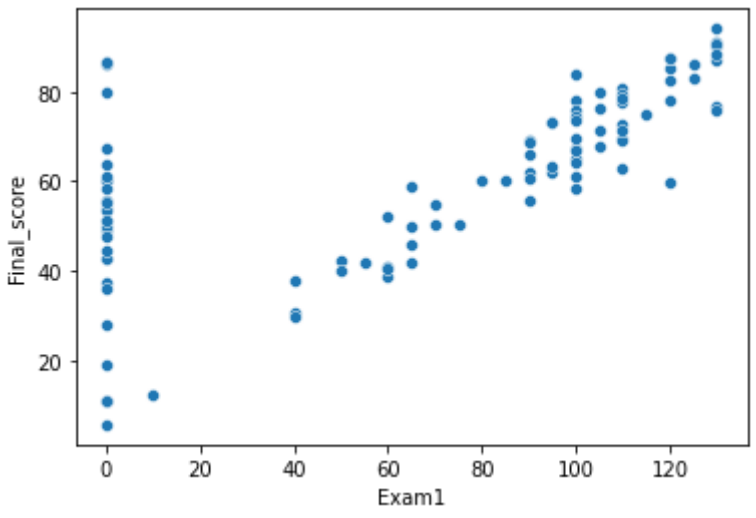
Out[54]:
```
<AxesSubplot:>
```



10.Visualise the relationship between Exam1 and Final score

In [55]:
```
sns.scatterplot(x='Exam1',y='Final_score',data=df_grade)
```

Out[55]:
```
<AxesSubplot:xlabel='Exam1', ylabel='Final_score'>
```



Import youtube top_500.csv

In [56]:
```
df_youtube=pd.read_csv('/home/sumon/Downloads/top_500.csv')
```

11. View the top 10 records

In [57]:
```
df_youtube.head(10)
```

Out[57]:

	Rank	Grade	Ch_name	Uploads	Subscriptions	Views
0	1st	A++	T-Series	14,297	135M	1,04,72,43,69,854
1	2nd	A++	Cocomelon - Nursery Rhymes	517	78.2M	57,05,42,90,512
2	3rd	A++	❀ Kids Diana Show	691	50.9M	24,15,76,78,368
3	4th	A++	Like Nastya	400	52.2M	30,59,12,57,306
4	5th	A++	SET India	37,017	69.3M	52,14,95,05,781
5	6th	A++	RRcherrypie	327	3.37M	2,33,88,29,659
6	7th	A++	Movieclips	35,226	36.3M	35,05,58,07,085
7	8th	A++	Zee TV	98,621	43M	41,54,42,59,461
8	9th	A++	Juguetes con Andre	691	5.17M	2,35,51,49,027
9	10th	A++	Vlad and Nikita	219	37.7M	18,07,40,55,450

12. Check the data types of the columns

In [58]:
```
df_youtube.dtypes
```

Out[58]:
```
Rank             object
Grade            object
Ch_name          object
Uploads          object
Subscriptions    object
Views            object
dtype: object
```

13. Print the missing values of each columns

In [59]:
```
df_youtube.isna().sum()
```

Out[59]:
```
Rank             0
Grade            5
Ch_name          0
Uploads          4
Subscriptions    0
Views            0
dtype: int64
```

14. Convert the columns uploads, Subcriptions and Views to numeric

In [60]:
```
df_youtube["Uploads"]=df_youtube["Uploads"].str.replace(",","")
```

In [61]:
```
df_youtube["Uploads"].head()
```

Out[61]:
```
0    14297
1      517
2      691
3      400
4    37017
Name: Uploads, dtype: object
```

In [62]:
```
df_youtube["Uploads"]=pd.to_numeric(df_youtube["Uploads"])
```

In [68]:
```
df_youtube["Subscriptions"]=df_youtube["Subscriptions"].str.extract('(\d+)'
```

In [69]:
```
df_youtube.Subscriptions=pd.to_numeric(df_youtube["Subscriptions"])
```

In [63]:
```
df_youtube["Views"]=df_youtube["Views"].str.replace(",","")
```

15. Check the rows which have '--' values in Views column

In [64]:
```
df_youtube[df_youtube["Views"]=="--"]["Views"]
```

Out[64]:
```
168    --
276    --
451    --
463    --
Name: Views, dtype: object
```

16. Replace the values by 0

In [65]:
```
df_youtube.loc[df_youtube.Views=="--","Views"]=0
```

In [66]:
```
df_youtube[df_youtube["Views"]=="--"]["Views"]
```

Out[66]:
```
Series([], Name: Views, dtype: object)
```

In [67]:
```
df_youtube["Views"]=pd.to_numeric(df_youtube["Views"])
```

In [70]:
```
df_youtube.dtypes
```

Out[70]:
```
Rank              object
Grade             object
Ch_name           object
Uploads          float64
Subscriptions      int64
Views              int64
dtype: object
```

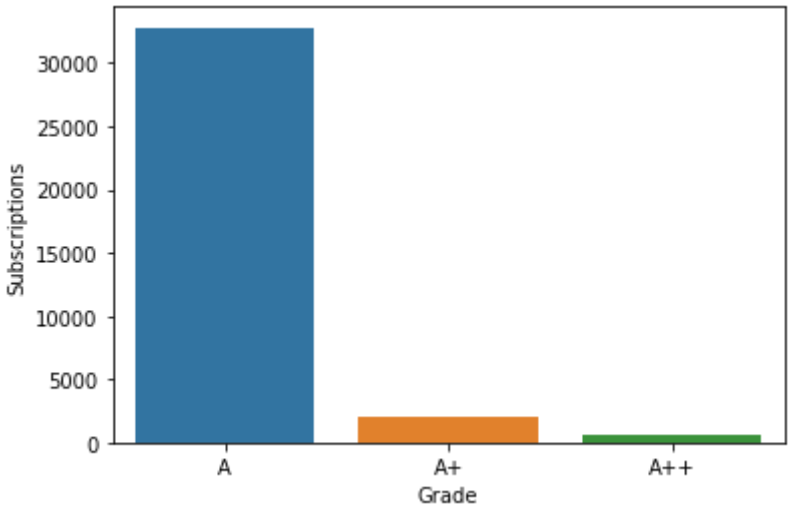
17. Print total channel's Subscription Gradewise

In [71]:
```
df_youtube.groupby("Grade")["Subscriptions"].sum()
```

Out[71]:
```
Grade
A      32801
A+      1963
A++      537
Name: Subscriptions, dtype: int64
```

In [72]:
```
sns.barplot(x=df_youtube.groupby("Grade")["Subscriptions"].sum().index,
            y=df_youtube.groupby("Grade")["Subscriptions"].sum(),
            data=df_youtube)
```

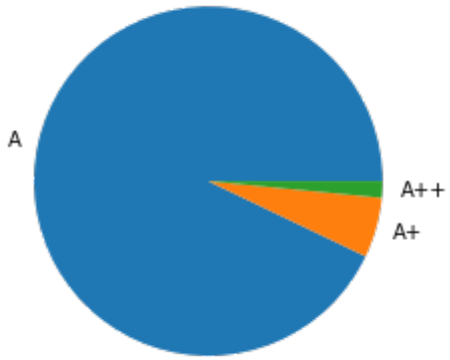
Out[72]:
```
<AxesSubplot:xlabel='Grade', ylabel='Subscriptions'>
```



18. Print a pie plot showing the total number of uploads grade-wise

In [73]:
```
plt.pie(df_youtube.groupby("Grade")["Subscriptions"].sum(),
        labels=df_youtube.groupby("Grade")["Subscriptions"].sum().index)
```

Out[73]:
```
([<matplotlib.patches.Wedge at 0x7fd6706eee50>,
  <matplotlib.patches.Wedge at 0x7fd6706fd400>,
  <matplotlib.patches.Wedge at 0x7fd6706fd8e0>],
 [Text(-1.0728871040966825, 0.24272054272977098, 'A'),
  Text(1.0600669833277905, -0.29369710733733495, 'A+'),
  Text(1.098744110987951, -0.0525488208173827, 'A++')])
```



Name : ***Sumon Singh***

Course : M.sc Computer Science With Specialization In Data Science

Paper : PSDS103 (Fundamentals Of Data Science)

Assignment : 3 (Sampling in R)

- **1. Load the Boston dataset :**

Code :

```
library(MASS)
df_Bostan=Boston
n=nrow(df_Bostan)
n
colnames(df_Bostan)
```

Output :

```
> library(MASS)
>
> df_Bostan=Boston
>
> n=nrow(df_Bostan)
> n
[1] 506
> colnames(df_Bostan)
 [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"     "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
>
```

2. Random sampling on Boston dataset :

Code :

```
s=sample(n,.8*n)
boston_tr=df_Bostan[s,]
boston_test=df_Bostan[-s,]
nrow(boston_tr)
nrow(boston_test)
```

Output :

```
> s=sample(n,.8*n)
> boston_tr=df_Bostan[s,]
> boston_test=df_Bostan[-s,]
> nrow(boston_tr)
[1] 404
> nrow(boston_test)
[1] 102
>
```

3. Convenience sampling on Boston dataset :

Code :

```
boston_tr1=df_Bostan[1:300,]
boston_test1=df_Bostan[301:506,]
nrow(boston_tr1)
nrow(boston_test1)
```

Ouput :

```
> boston_tr1=df_Bostan[1:300,]
> boston_test1=df_Bostan[301:506,]
> nrow(boston_tr1)
[1] 300
> nrow(boston_test1)
[1] 206
```

4. Systematic sampling on Boston dataset :

Code :

```
df_boston_order=df_Bostan[order(-df_Bostan$medv),]
row_sel=seq(from=1,to=506,by=2)
boston_tr2=df_Bostan[row_sel,]
boston_test2=df_Bostan[-row_sel,]
nrow(boston_tr2)
nrow(boston_test2)
```

Output :

```
> df_boston_order=df_Bostan[order(-df_Bostan$medv),]
> row_sel=seq(from=1,to=506,by=2)
> boston_tr2=df_Bostan[row_sel,]
> boston_test2=df_Bostan[-row_sel,]
> nrow(boston_tr2)
[1] 253
> nrow(boston_test2)
[1] 253
```

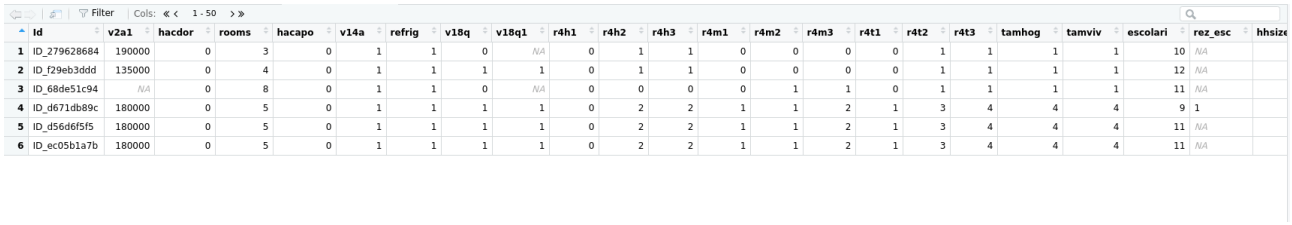
5. Stratified sampling on income dataset :

A. Load income dataset --

Code :

```
df_income=read.csv('/home/sumon/Downloads/income_tr.csv')
View(head(df_income))
table(df_income$Target)
```

Output :



	Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	r4h2	r4h3	r4m1	r4m2	r4m3	r4t1	r4t2	r4t3	tamhog	tamviv	escolari	rez_esc	hhsize
1	ID_279628684	190000	0	3	0	1	1	0	NA	0	1	1	0	0	0	0	1	1	1	1	10	NA	
2	ID_f29eb3ddd	135000	0	4	0	1	1	1	1	0	1	1	0	0	0	0	1	1	1	1	12	NA	
3	ID_68de51c94	NA	0	8	0	1	1	0	NA	0	0	0	0	1	1	0	1	1	1	1	11	NA	
4	ID_d671db89c	180000	0	5	0	1	1	1	1	0	2	2	1	1	2	1	3	4	4	4	9	1	
5	ID_d56d6f5f5	180000	0	5	0	1	1	1	1	0	2	2	1	1	2	1	3	4	4	4	11	NA	
6	ID_ec05b1a7b	180000	0	5	0	1	1	1	1	0	2	2	1	1	2	1	3	4	4	4	11	NA	

```
> df_income=read.csv('/home/sumon/Downloads/income_tr.csv')
> View(head(df_income))
> table(df_income$Target)

   1    2    3    4
 755 1597 1209 5996
```

B. Making groups or strata --

Code :

```
strata_1=df_income[df_income$Target==1,]
strata_2=df_income[df_income$Target==2,]
strata_3=df_income[df_income$Target==3,]
strata_4=df_income[df_income$Target==4,]
nrow(strata_1)
nrow(strata_2)
nrow(strata_3)
nrow(strata_4)
```

Output :

```
> strata_1=df_income[df_income$Target==1,]
> strata_2=df_income[df_income$Target==2,]
> strata_3=df_income[df_income$Target==3,]
> strata_4=df_income[df_income$Target==4,]
> nrow(strata_1)
[1] 755
> nrow(strata_2)
[1] 1597
> nrow(strata_3)
[1] 1209
> nrow(strata_4)
[1] 5996
```

C. Splitting the strata into training and testing sets :

Code :

```
s1=sample(nrow(strata_1),.8*nrow(strata_1))

s1_tr=strata_1[s1,]

s1_test=strata_1[-s1,]

nrow(s1_test)

nrow(s1_tr)

s2=sample(nrow(strata_2),.8*nrow(strata_2))

s2_tr=strata_2[s2,]

s2_test=strata_2[-s2,]

nrow(s2_test)

nrow(s2_tr)

s3=sample(nrow(strata_3),.8*nrow(strata_3))

s3_tr=strata_3[s3,]
```

```
s3_test=strata_3[-s3,]

nrow(s3_test)

nrow(s3_tr)

s4=sample(nrow(strata_4),.8*nrow(strata_4))

s4_tr=strata_4[s4,]

s4_test=strata_4[-s4,]

nrow(s4_test)

nrow(s4_tr)
```

Output :

```
> s1=sample(nrow(strata_1),.8*nrow(strata_1))
> s1_tr=strata_1[s1,]
> s1_test=strata_1[-s1,]
> nrow(s1_test)
[1] 151
> nrow(s1_tr)
[1] 604
> s2=sample(nrow(strata_2),.8*nrow(strata_2))
> s2_tr=strata_2[s2,]
> s2_test=strata_2[-s2,]
> nrow(s2_test)
[1] 320
> nrow(s2_tr)
[1] 1277
> s3=sample(nrow(strata_3),.8*nrow(strata_3))
> s3_tr=strata_3[s3,]
> s3_test=strata_3[-s3,]
> nrow(s3_test)
[1] 242
> nrow(s3_tr)
[1] 967
> s4=sample(nrow(strata_4),.8*nrow(strata_4))
> s4_tr=strata_4[s4,]
> s4_test=strata_4[-s4,]
> nrow(s4_test)
[1] 1200
> nrow(s4_tr)
[1] 4796
```

D. Combine all the test and training strata :

Code :

```
income_tr=rbind(s1_tr,s2_tr,s3_tr,s4_tr)

income_test=rbind(s1_test,s2_test,s3_test,s4_test)

nrow(income_tr)

nrow(income_test)
```

Output :

```
> income_tr=rbind(s1_tr,s2_tr,s3_tr,s4_tr)
> income_test=rbind(s1_test,s2_test,s3_test,s4_test)
> nrow(income_tr)
[1] 7644
> nrow(income_test)
[1] 1913
```

Name : Sumon Singh

Course : M.sc Computer Science With Specialization In Data Science

Paper : PSDS103 (Fundamentals Of Data Science)

Assignment : 4 (MySql)

Create Database :

Command :

```
create database library;
use library;
```

Create Tables :

Command :

```
create table book(title varchar(30) primary key,
author varchar(30),
cost int,
publisher_name varchar(30));

create table book_Status(accession_no varchar(30) primary key,
title varchar(30),
foreign key(title) references book(title),
status varchar(30));

create table member_master(
member_id varchar(30) primary key,
mem_name varchar(30),
mem_type varchar(30),
location varchar(30),
city varchar(30),
contact_no bigint,
no_books_issued int);

create table issue_return(
txn_id varchar(10) primary key,
date_of_issue date,
due_date date,
member_id varchar(30),
foreign key(member_id) references member_master(member_id),
accession_no varchar(30),
foreign key(accession_no) references book_Status(accession_no),
date_of_return date);
```

Insert records :

```
insert into book values("beginners programming in C++","Balaguruswamy",500,"TMH");
insert into book values("Let us C","Kanethkar",500,"TMH");

insert into book_Status values("B001","Let us C","available");
insert into book_Status values("B002","Let us C","issued");

insert into book values('programming with Java','Balaguruswamy',350,'TMH');
insert into book values('ASP.NET 3.5','Anne Boehm',600,'phi');
insert into book values('Beginners VB','Julia case bradley',600,'TMH');
```

Quries :

A. select title,author from book;

command : select title,author from book;

title	author
ASP.NET 3.5	Anne Boehm
beginners programming in C++	Balaguruswamy
Beginners VB	Julia case bradley
Da vinci code	Dan Brown
Let us C	Kanethkar
programming with Java	Balaguruswamy

B. select title,cost from book;

command : select title,cost from book ;

title	cost
ASP.NET 3.5	600
beginners programming in C++	500
Beginners VB	600
Da vinci code	1000
Let us C	500
programming with Java	350

C. print the details of the books which are either C/ C++

command : select * from book where title like '%C%' or title like '%C++%';

title	author	cost	publisher_name
beginners programming in C++	Balaguruswamy	500	TMH
Da vinci code	Dan Brown	1000	corgi
Let us C	Kanethkar	500	TMH

D. print the details of the books written by Balaguruswamy

command : select * from book where author ="Balaguruswamy";

title	author	cost	publisher_name
beginners programming in C++	Balaguruswamy	500	TMH
programming with Java	Balaguruswamy	350	TMH

E. print the title, author and publisher name of the books related to programming

command : select title,author,publisher_name from book where title like "%programming%";

title	author	publisher_name
beginners programming in C++	Balaguruswamy	TMH
programming with Java	Balaguruswamy	TMH

F. select the books where price ranges between 200 and 600

command : select * from book where cost >=200 and cost <=600;

title	author	cost	publisher_name
ASP.NET 3.5	Anne Boehm	600	phi
beginners programming in C++	Balaguruswamy	500	TMH
Beginners VB	Julia case bradley	600	TMH
Let us C	Kanethkar	500	TMH
programming with Java	Balaguruswamy	350	TMH

G. select the books where the author is not balaguruswamy

command : select * from book where author != "Balaguruswamy";

title	author	cost	publisher_name
ASP.NET 3.5	Anne Boehm	600	phi
Beginners VB	Julia case bradley	600	TMH
Da vinci code	Dan Brown	1000	corgi
Let us C	Kanethkar	500	TMH

H. select the books written by either Balguruswamy or Kanethkar or Anne Boehm

command : select * from book where author in ("Balaguruswamy","Anne Boehm","Julia case bradley");

title	author	cost	publisher_name
ASP.NET 3.5	Anne Boehm	600	phi
beginners programming in C++	Balaguruswamy	500	TMH
Beginners VB	Julia case bradley	600	TMH
programming with Java	Balaguruswamy	350	TMH

I. sort the table based on cost in descending order

command : select * from book order by(cost) desc;

title	author	cost	publisher_name
Da vinci code	Dan Brown	1000	corgi
ASP.NET 3.5	Anne Boehm	600	phi
Beginners VB	Julia case bradley	600	TMH
beginners programming in	Balaguruswamy	500	TMH

C++

Let us C	Kanethkar	500 TMH
programming with Java	Balaguruswamy	350 TMH

Create Tables :

Command :

```
create table product( product_id varchar(30) primary key,
product_name varchar(100),
company_name varchar(100),
unit_price int);

create table order_prod( order_id varchar(30) primary key,
product_id varchar(30),
total_units int,
customer varchar(100),
```

Insert Records :

```
insert into product values ('100','shampoo','panteen',180);
insert into product values ('101','dendorant','park avenue',400);
insert into product values ('102','tooth paste','colgate',150);
insert into product values ('103','soap','lux',75);
insert into product values ('104','hair gel','laureal',300);

insert into order_prod values('o1','101',30,'lifestyle');
insert into order_prod values('o2','101',5,'shoppers shop');
insert into order_prod values('o3','103',25,'spencer');
insert into order_prod values('o4','101',10,'food bazaar');
insert into order_prod values('o5','103',200,'big bazar');
```

Tables :

```
select * from product;
```

product_id	product_name	company_name	unit_price
100	shampoo	panteen	180
101	dendorant	park avenue	400
102	tooth paste	colgate	150
103	soap	lux	75
104	hair gel	laureal	300

```
select * from order_prod;
```

order_id	product_id	total_units	customer
o1	101	30	lifestyle
o2	101	5	shoppers shop
o3	103	25	spencer
o4	101	10	food bazaar

o5 103 200 big bazar

J. Display the name and price of the products which are ordered by Big Bazaar

command : select product_name,unit_price from product inner join order_prod on
product.product_id=order_prod.product_id
where customer='big bazar';

product_name unit_price
soap 75

K. Get the names of the Customers who have placed orders for Products having price more than 200 per unit.

command : select customer from product inner join order_prod on
product.product_id=order_prod.product_id where unit_price > 200;

customer
lifestyle
shoppers shop
food bazaar

L. Get the Product_name and company_name of the Products ordered by Food Bazaar.

command : select product_name,company_name from product inner join order_prod on
product.product_id=order_prod.product_id where customer='food bazaar';

product_name company_name
dendorant park avenue

M. print the product which has the maximum price

command : select * from product where unit_price = (select max(unit_price) from product);

product_id product_name company_name unit_price

101 dendorant park avenue 400

N. print the product name and the total quantity which has been ordered

command : select product_name,sum(total_units) from product inner join order_prod on product.product_id=order_prod.product_id group by order_prod.product_id;

product_name	sum(total_units)
dendorant	45
soap	225

O. To select the product _ids of the products which have been ordered at least once?

command : select product_id from order_prod group by product_id having count(product_id)>=1;

product_id
101
103

Name : Sumon Singh

Course : M.sc Computer Science With Specialization In Data Science

Paper : PSDS103 (Fundamentals Of Data Science)

Assignment : 4 (MySql)

In [1]:
```
import pymysql
```

In [2]:
```
con=pymysql.connect(user="root",password="root",database="library",host="l
```

In [3]:
```
c1=con.cursor()
```

In [4]:
```
sql="describe member_master"
```

In [5]:
```
c1.execute(sql)
```

Out[5]:
```
7
```

In [6]:
```
c1.fetchall()
```

Out[6]:
```
(('member_id', 'varchar(30)', 'NO', 'PRI', None, ''),
 ('mem_name', 'varchar(30)', 'YES', '', None, ''),
 ('mem_type', 'varchar(30)', 'YES', '', None, ''),
 ('location', 'varchar(30)', 'YES', '', None, ''),
 ('city', 'varchar(30)', 'YES', '', None, ''),
 ('contact_no', 'bigint', 'YES', '', None, ''),
 ('no_books_issued', 'int', 'YES', '', None, ''))
```

In [7]:
```
sql1="show tables"
```

In [8]:
```
c1.execute(sql1)
```

Out[8]:
```
6
```

In [9]:
```
rs1=c1.fetchall()
rs1
```

Out[9]:
```
(('book',),
 ('book_Status',),
 ('issue_return',),
 ('member_master',),
 ('order_prod',),
 ('product',))
```

In [10]:
```
sql2="select * from book"
```

In [11]:
```
c1.execute(sql2)
```

Out[11]:
```
6
```

In [12]:
```
rs1=c1.fetchall()
rs1
```

Out[12]:
```
(('ASP.NET 3.5', 'Anne Boehm', 600, 'phi'),
 ('beginners programming in C++', 'Balaguruswamy', 500, 'TMH'),
 ('Beginners VB', 'Julia case bradley', 600, 'TMH'),
 ('Da vinci code', 'Dan Brown', 1000, 'corgi'),
 ('Let us C', 'Kanethkar', 500, 'TMH'),
 ('programming with Java', 'Balaguruswamy', 350, 'TMH'))
```

In [20]:
```
t=input('Enter title of the book?')
a=input('Enter author of the book?')
c=int(input('enter cost of the book?'))
p=input('Enter the publisher name?')

sql_insert="insert into book values('{0}','{1}','{2}','{3}')".format(t,a,c
```

```
Enter title of the book?Da vinci code
Enter author of the book?Dan Brown
enter cost of the book?600
Enter the publisher name?corgi
```

In [21]:
```
sql_insert
```

Out[21]:
```
"insert into book values('Da vinci code','Dan Brown','600','corgi')"
```

In [22]:
```
c1.execute(sql_insert)
```

Out[22]:
```
1
```

In [23]:
```
c1.execute("commit")
```

Out[23]:
```
0
```

In [26]:
```
# update the cost of the book from 600 to 500
tname=input('Enter table name?')
cname=input('Enter column name?')
val=int(input('Enter the new value?'))

sql_update="update {0} set {1}={2} where title='Da vinci code'".format(tnar
```

```
Enter table name?book
Enter column name?cost
Enter the new value?1000
```

In [27]:
```
sql_update
```

Out[27]:
```
"update book set cost=1000 where title='Da vinci code'"
```

In [28]:
```
c1.execute(sql_update)
```

Out[28]:
```
1
```

In [29]:
```
c1.execute("commit")
```

Out[29]:
```
0
```

Name : Sumon Singh

Course : M.sc Computer Science With Specialization In
Data Science

Paper : PSDS103 (Fundamentals Of Data Science)

Assignment : 5 (Data curation using MongoDB)

```
> show dbs;
admin    0.000GB
config   0.000GB
lib      0.000GB
library  0.000GB
local    0.000GB
```

Note : The Database name is Books and the collection on which all operations are performed is book_details.

1. Change current database :

Code : use Books ; //Books is the name of the databse

Output :

```
> use Books;
switched to db Books
```

2. Create a collection book_details

Code : db.createCollection("book_details")

Output :

```
> db.createCollection("book_details")
{ "ok" : 1 }
```

3. Insert more documents in book collection

Code :

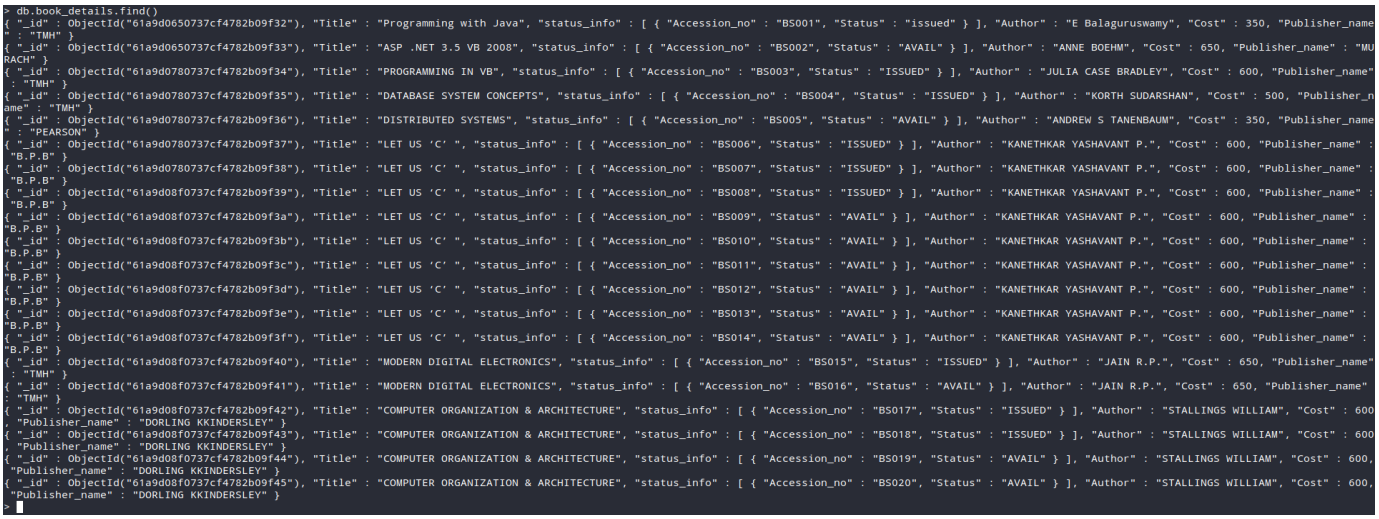
```
> db.book_details.insert({"Title":"Programming with Java","status_info":[{"Accession_no":"BS001","Status":"issued"}],"Author":"E Balaguruswamy","Cost":350,"Publisher_name":"TMH"})
WriteResult({ "nInserted" : 1 })
>
> db.book_details.insert({"Title":"ASP .NET 3.5 VB 2008","status_info":[{"Accession_no":"BS002","Status":"AVAIL"}],"Author":"ANNE BOEHM","Cost":650,"Publisher_name":"MURACH"})
WriteResult({ "nInserted" : 1 })
> db.book_details.insert({"Title":"PROGRAMMING IN VB","status_info":[{"Accession_no":"BS003","Status":"ISSUED"}],"Author":"JULIA CASE BRADLEY","Cost":600,"Publisher_name":"TMH"})
WriteResult({ "nInserted" : 1 })
>
> db.book_details.insert({"Title":"DATABASE SYSTEM CONCEPTS","status_info":[{"Accession_no":"BS004","Status":"ISSUED"}],"Author":"KORTH SUDARSHAN","Cost":500,"Publisher_name":"TMH"})
WriteResult({ "nInserted" : 1 })
>
> db.book_details.insert({"Title":"DISTRIBUTED SYSTEMS","status_info":[{"Accession_no":"BS005","Status":"AVAIL"}],"Author":"ANDREW S TANENBAUM","Cost":350,"Publisher_name":"PEARSON"})
WriteResult({ "nInserted" : 1 })
>
> db.book_details.insert({"Title":"LET US 'C' ","status_info":[{"Accession_no":"BS006","Status":"ISSUED"}],"Author":"KANETHKAR YASHAVANT P.","Cost":600,"Publisher_name":"B.P.B"})
WriteResult({ "nInserted" : 1 })
>
> db.book_details.insert({"Title":"LET US 'C' ","status_info":[{"Accession_no":"BS007","Status":"ISSUED"}],"Author":"KANETHKAR YASHAVANT P.","Cost":600,"Publisher_name":"B.P.B"})
WriteResult({ "nInserted" : 1 })
> db.book_details.insert({"Title":"LET US 'C' ","status_info":[{"Accession_no":"BS008","Status":"ISSUED"}],"Author":"KANETHKAR YASHAVANT P.","Cost":600,"Publisher_name":"B.P.B"})
WriteResult({ "nInserted" : 1 })
>
> db.book_details.insert({"Title":"LET US 'C' ","status_info":[{"Accession_no":"BS009","Status":"AVAIL"}],"Author":"KANETHKAR YASHAVANT P.","Cost":600,"Publisher_name":"B.P.B"})
WriteResult({ "nInserted" : 1 })
>
> db.book_details.insert({"Title":"LET US 'C' ","status_info":[{"Accession_no":"BS010","Status":"AVAIL"}],"Author":"KANETHKAR YASHAVANT P.","Cost":600,"Publisher_name":"B.P.B"})
WriteResult({ "nInserted" : 1 })
>
> db.book_details.insert({"Title":"LET US 'C' ","status_info":[{"Accession_no":"BS011","Status":"AVAIL"}],"Author":"KANETHKAR YASHAVANT P.","Cost":600,"Publisher_name":"B.P.B"})
WriteResult({ "nInserted" : 1 })
>
> db.book_details.insert({"Title":"LET US 'C' ","status_info":[{"Accession_no":"BS012","Status":"AVAIL"}],"Author":"KANETHKAR YASHAVANT P.","Cost":600,"Publisher_name":"B.P.B"})
WriteResult({ "nInserted" : 1 })
>
> db.book_details.insert({"Title":"LET US 'C' ","status_info":[{"Accession_no":"BS013","Status":"AVAIL"}],"Author":"KANETHKAR YASHAVANT P.","Cost":600,"Publisher_name":"B.P.B"})
WriteResult({ "nInserted" : 1 })
>
> db.book_details.insert({"Title":"LET US 'C' ","status_info":[{"Accession_no":"BS014","Status":"AVAIL"}],"Author":"KANETHKAR YASHAVANT P.","Cost":600,"Publisher_name":"B.P.B"})
WriteResult({ "nInserted" : 1 })
>
> db.book_details.insert({"Title":"MODERN DIGITAL ELECTRONICS","status_info":[{"Accession_no":"BS015","Status":"ISSUED"}],"Author":"JAIN R.P.","Cost":650,"Publisher_name":"TMH"})
WriteResult({ "nInserted" : 1 })
>
> db.book_details.insert({"Title":"MODERN DIGITAL ELECTRONICS","status_info":[{"Accession_no":"BS016","Status":"AVAIL"}],"Author":"JAIN R.P.","Cost":650,"Publisher_name":"TMH"})
WriteResult({ "nInserted" : 1 })
>
> db.book_details.insert({"Title":"COMPUTER ORGANIZATION & ARCHITECTURE","status_info":[{"Accession_no":"BS017","Status":"ISSUED"}],"Author":"STALLINGS WILLIAM","Cost":600,"Publisher_name":"DORLING KKINDERSLEY"})
WriteResult({ "nInserted" : 1 })
>
> db.book_details.insert({"Title":"COMPUTER ORGANIZATION & ARCHITECTURE","status_info":[{"Accession_no":"BS018","Status":"ISSUED"}],"Author":"STALLINGS WILLIAM","Cost":600,"Publisher_name":"DORLING KKINDERSLEY"})
WriteResult({ "nInserted" : 1 })
>
```

```
> db.book_details.insert({"Title":"COMPUTER ORGANIZATION & ARCHITECTURE","status_info":[{"Accession_no":"BS019","Status":"AVAIL"}],"Author":"STALLINGS WILLIAM","Cost":600,"Publisher_name":"DORLING KKINDERSLEY"})
WriteResult({ "nInserted" : 1 })
>
> db.book_details.insert({"Title":"COMPUTER ORGANIZATION & ARCHITECTURE","status_info":[{"Accession_no":"BS020","Status":"AVAIL"}],"Author":"STALLINGS WILLIAM","Cost":600,"Publisher_name":"DORLING KKINDERSLEY"})
WriteResult({ "nInserted" : 1 })
>
```

4. To select all records in book collection

Code : db.book_details.find()

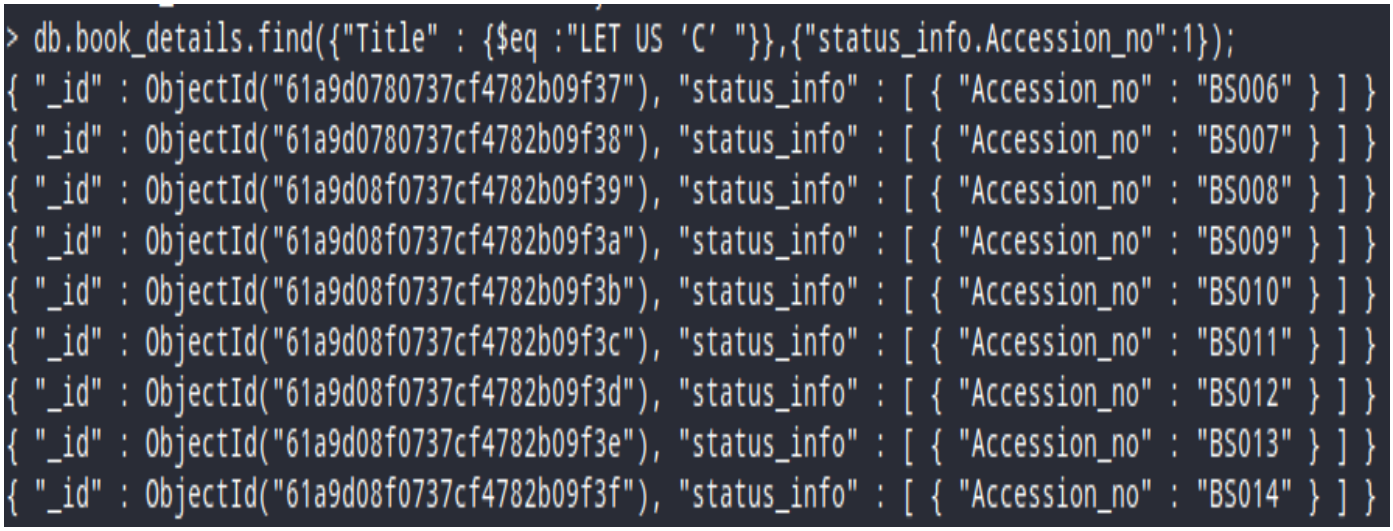
Output :



5. To print the accession no of books where title is Let us C

Code : db.book_details.find({"Title" : {$eq :"LET US 'C' "}},{"status_info.Accession_no":1})

Output :



6. To print the accession no of all books

Code : db.book_details.find({},{"status_info.Accession_no":1});

Output :

```
> db.book_details.find({},{"status_info.Accession_no":1});
{ "_id" : ObjectId("61a9d0650737cf4782b09f32"), "status_info" : [ { "Accession_no" : "BS001" } ] }
{ "_id" : ObjectId("61a9d0650737cf4782b09f33"), "status_info" : [ { "Accession_no" : "BS002" } ] }
{ "_id" : ObjectId("61a9d0780737cf4782b09f34"), "status_info" : [ { "Accession_no" : "BS003" } ] }
{ "_id" : ObjectId("61a9d0780737cf4782b09f35"), "status_info" : [ { "Accession_no" : "BS004" } ] }
{ "_id" : ObjectId("61a9d0780737cf4782b09f36"), "status_info" : [ { "Accession_no" : "BS005" } ] }
{ "_id" : ObjectId("61a9d0780737cf4782b09f37"), "status_info" : [ { "Accession_no" : "BS006" } ] }
{ "_id" : ObjectId("61a9d0780737cf4782b09f38"), "status_info" : [ { "Accession_no" : "BS007" } ] }
{ "_id" : ObjectId("61a9d08f0737cf4782b09f39"), "status_info" : [ { "Accession_no" : "BS008" } ] }
{ "_id" : ObjectId("61a9d08f0737cf4782b09f3a"), "status_info" : [ { "Accession_no" : "BS009" } ] }
{ "_id" : ObjectId("61a9d08f0737cf4782b09f3b"), "status_info" : [ { "Accession_no" : "BS010" } ] }
{ "_id" : ObjectId("61a9d08f0737cf4782b09f3c"), "status_info" : [ { "Accession_no" : "BS011" } ] }
{ "_id" : ObjectId("61a9d08f0737cf4782b09f3d"), "status_info" : [ { "Accession_no" : "BS012" } ] }
{ "_id" : ObjectId("61a9d08f0737cf4782b09f3e"), "status_info" : [ { "Accession_no" : "BS013" } ] }
{ "_id" : ObjectId("61a9d08f0737cf4782b09f3f"), "status_info" : [ { "Accession_no" : "BS014" } ] }
{ "_id" : ObjectId("61a9d08f0737cf4782b09f40"), "status_info" : [ { "Accession_no" : "BS015" } ] }
{ "_id" : ObjectId("61a9d08f0737cf4782b09f41"), "status_info" : [ { "Accession_no" : "BS016" } ] }
{ "_id" : ObjectId("61a9d08f0737cf4782b09f42"), "status_info" : [ { "Accession_no" : "BS017" } ] }
{ "_id" : ObjectId("61a9d08f0737cf4782b09f43"), "status_info" : [ { "Accession_no" : "BS018" } ] }
{ "_id" : ObjectId("61a9d08f0737cf4782b09f44"), "status_info" : [ { "Accession_no" : "BS019" } ] }
{ "_id" : ObjectId("61a9d08f0737cf4782b09f45"), "status_info" : [ { "Accession_no" : "BS020" } ] }
```

7. To print the title of the books with accession_no BS001

Code : db.book_details.find({"status_info.Accession_no":"BS001"},{"Title":1});

Output :

{ "_id" : ObjectId("61a7c0d1f2ab226465310775"), "Title" : "Programming with Java" }

8. To print the total copies of the books which are authored by Balaguruswamy which are available

Code : db.book_details.find({“Author”: "E Balaguruswamy","status_info.Status":"AVAIL”}).count();

Output :

>db.book_details.find({“Author”: "E Balaguruswamy","status_info.Status":"AVAIL”}).count();

0

9. Update query book “let us C” with cost 1000

Code : db.book_details.update({"Title":"LET US ‘C’ "},{$set :{"Cost" : 1000}}, {"multi":true});

Output :

> db.book_details.update({"Title":"LET US ‘C’ "},{$set :{"Cost" : 1000}}, {"multi":true});

WriteResult({ "nMatched" : 9, "nUpserted" : 0, "nModified" : 9 })

> db.book_details.find({"Title" :"LET US ‘C’ "},{Cost:1})

{ "_id" : ObjectId("61a9d0780737cf4782b09f37"), "Cost" : 1000 }

{ "_id" : ObjectId("61a9d0780737cf4782b09f38"), "Cost" : 1000 }

{ "_id" : ObjectId("61a9d08f0737cf4782b09f39"), "Cost" : 1000 }

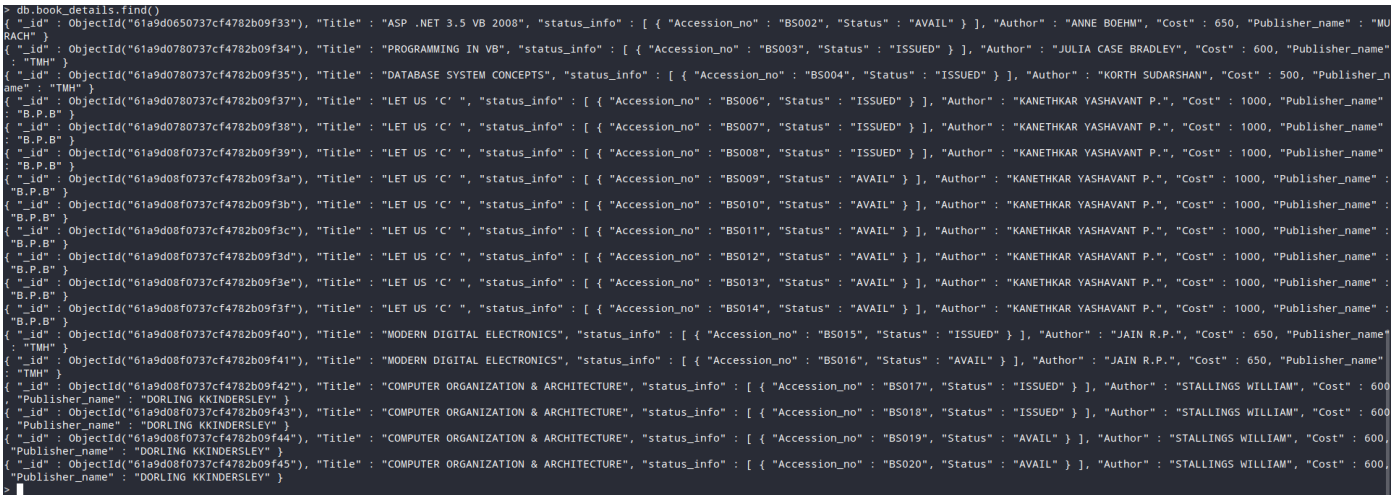
{ "_id" : ObjectId("61a9d08f0737cf4782b09f3a"), "Cost" : 1000 }
{ "_id" : ObjectId("61a9d08f0737cf4782b09f3b"), "Cost" : 1000 }
{ "_id" : ObjectId("61a9d08f0737cf4782b09f3c"), "Cost" : 1000 }
{ "_id" : ObjectId("61a9d08f0737cf4782b09f3d"), "Cost" : 1000 }
{ "_id" : ObjectId("61a9d08f0737cf4782b09f3e"), "Cost" : 1000 }
{ "_id" : ObjectId("61a9d08f0737cf4782b09f3f"), "Cost" : 1000 }

10. Delete specific documents

Code : db.book_details.remove({"Cost":350}) // Remove books having cost 350

Output :

> db.book_details.remove({"Cost":350})
WriteResult({ "nRemoved" : 2 })
> db.book_details.find()



11. db.collection.deleteMany()

Code : db.book_details.deleteMany({"status_info.Status":"AVAIL"})

Output :

> db.book_details.deleteMany({"status_info.Status":"AVAIL"}) // Delete books with Status AVAIL

{ "acknowledged" : true, "deletedCount" : 10 }

12. Delete all documents

Code : db.book_details.remove({})

Output :

> db.book_details.remove({})

WriteResult({ "nRemoved" : 8 })
> db.book_details.find({})
>

Linear Regression On Boston_Dataset,Assignment-6

Sumon Singh,M.Sc Computer Science Specialization in Data Science

19/12/2021

Linear Regression on Boston Dataset

This is a r-script which performs linear regression on Boston dataset. The script contains two models. The first model is a simple linear regression which evaluates the medv values based on the feature lstat. The sceond model is a multiple linear regression which evaluates medv values based on all the featues of the dataset. The goodness of each models have been evaluated using the r2-score,mean-absolute-error,root-mean-square-error and min-max accuarcy.

Load required libraries

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(Metrics)
```

Read Dataset

```
df = read.csv('/home/sumon/Desktop/Datasets/boston_dataset.csv')
head(df)
```

```
##   X    crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 1 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
## 4 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
## 5 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
## 6 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
##   medv
## 1 24.0
## 2 21.6
## 3 34.7
## 4 33.4
## 5 36.2
## 6 28.7
```

```
dim(df)
```

```
## [1] 506  15
```

```
str(df)
```

```
## 'data.frame':    506 obs. of  15 variables:
##  $ X      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : int  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

```
df = df[,-1]
data.frame(sapply(df, function(x)(is.null(x))))
```

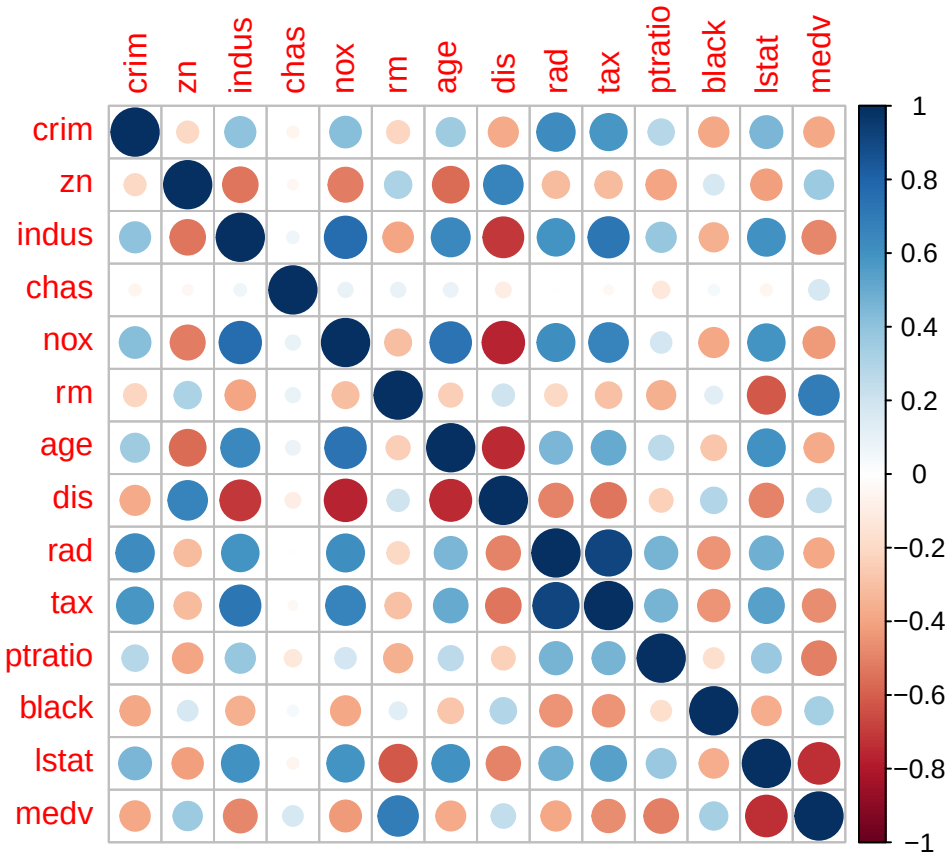
```
##         sapply.df..function.x...is.null.x...
## crim                                   FALSE
## zn                                     FALSE
## indus                                  FALSE
## chas                                   FALSE
## nox                                    FALSE
## rm                                     FALSE
## age                                    FALSE
## dis                                    FALSE
## rad                                    FALSE
## tax                                    FALSE
```

```
## ptratio                                 FALSE
## black                                   FALSE
## lstat                                   FALSE
## medv                                    FALSE
```

```
mat = cor(df)
mat
```

```
##                crim          zn       indus         chas         nox
## crim     1.00000000 -0.20046922  0.40658341 -0.055891582  0.42097171
## zn      -0.20046922  1.00000000 -0.53382819 -0.042696719 -0.51660371
## indus    0.40658341 -0.53382819  1.00000000  0.062938027  0.76365145
## chas    -0.05589158 -0.04269672  0.06293803  1.000000000  0.09120281
## nox      0.42097171 -0.51660371  0.76365145  0.091202807  1.00000000
## rm      -0.21924670  0.31199059 -0.39167585  0.091251225 -0.30218819
## age      0.35273425 -0.56953734  0.64477851  0.086517774  0.73147010
## dis     -0.37967009  0.66440822 -0.70802699 -0.099175780 -0.76923011
## rad      0.62550515 -0.31194783  0.59512927 -0.007368241  0.61144056
## tax      0.58276431 -0.31456332  0.72076018 -0.035586518  0.66802320
## ptratio  0.28994558 -0.39167855  0.38324756 -0.121515174  0.18893268
## black   -0.38506394  0.17552032 -0.35697654  0.048788485 -0.38005064
## lstat    0.45562148 -0.41299457  0.60379972 -0.053929298  0.59087892
## medv    -0.38830461  0.36044534 -0.48372516  0.175260177 -0.42732077
##                  rm         age         dis          rad         tax    ptratio
## crim    -0.21924670  0.35273425 -0.37967009  0.625505145  0.58276431  0.2899456
## zn       0.31199059 -0.56953734  0.66440822 -0.311947826 -0.31456332 -0.3916785
## indus   -0.39167585  0.64477851 -0.70802699  0.595129275  0.72076018  0.3832476
## chas     0.09125123  0.08651777 -0.09917578 -0.007368241 -0.03558652 -0.1215152
## nox     -0.30218819  0.73147010 -0.76923011  0.611440563  0.66802320  0.1889327
## rm       1.00000000 -0.24026493  0.20524621 -0.209846668 -0.29204783 -0.3555015
## age     -0.24026493  1.00000000 -0.74788054  0.456022452  0.50645559  0.2615150
## dis      0.20524621 -0.74788054  1.00000000 -0.494587930 -0.53443158 -0.2324705
## rad     -0.20984667  0.45602245 -0.49458793  1.000000000  0.91022819  0.4647412
## tax     -0.29204783  0.50645559 -0.53443158  0.910228189  1.00000000  0.4608530
## ptratio -0.35550149  0.26151501 -0.23247054  0.464741179  0.46085304  1.0000000
## black    0.12806864 -0.27353398  0.29151167 -0.444412816 -0.44180801 -0.1773833
## lstat   -0.61380827  0.60233853 -0.49699583  0.488676335  0.54399341  0.3740443
## medv     0.69535995 -0.37695457  0.24992873 -0.381626231 -0.46853593 -0.5077867
##                black      lstat       medv
## crim    -0.38506394  0.4556215 -0.3883046
## zn       0.17552032 -0.4129946  0.3604453
## indus   -0.35697654  0.6037997 -0.4837252
## chas     0.04878848 -0.0539293  0.1752602
## nox     -0.38005064  0.5908789 -0.4273208
## rm       0.12806864 -0.6138083  0.6953599
## age     -0.27353398  0.6023385 -0.3769546
## dis      0.29151167 -0.4969958  0.2499287
## rad     -0.44441282  0.4886763 -0.3816262
## tax     -0.44180801  0.5439934 -0.4685359
## ptratio -0.17738330  0.3740443 -0.5077867
## black    1.00000000 -0.3660869  0.3334608
## lstat   -0.36608690  1.0000000 -0.7376627
## medv     0.33346082 -0.7376627  1.0000000
```

Plot



Random Sampling

```
s = sample(nrow(df),0.8*nrow(df))
df_tr = df[s,]
dim(df_tr)
```

```
## [1] 404  14
```

```
df_test  = df[-s,]
dim(df_test)
```

```
## [1] 102  14
```

Model1 (Simple Linear Regression)

```
cor(df$lstat,df$medv)
```

```
## [1] -0.7376627
```

```
model1 = lm(medv~lstat,data = df_tr)
summary(model1)
```

```
##
## Call:
## lm(formula = medv ~ lstat, data = df_tr)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.076  -4.009  -1.400   2.029  24.556
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.29115    0.62897   54.52   <2e-16 ***
## lstat       -0.92831    0.04326  -21.46   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.325 on 402 degrees of freedom
## Multiple R-squared:  0.534,  Adjusted R-squared:  0.5328
## F-statistic: 460.6 on 1 and 402 DF,  p-value: < 2.2e-16
```

```
p1 = predict(model1,df_test)
r2_sq = cor(df_test$medv,p1)**2
r2_sq
```

```
## [1] 0.593742
```

```
ac_pred = data.frame('Actual'=df_test$medv,'Predicted'=p1)
min_max_accuracy=mean(apply(ac_pred, 1, min)/apply(ac_pred, 1, max))
min_max_accuracy
```

```
## [1] 0.8399994
```

```
mae(df_test$medv,p1)
```

```
## [1] 4.01241
```

```
rmse(df_test$medv,p1)
```

```
## [1] 5.775634
```

Model2 (Mltiple Linear Regression)

```
corr_target=data.frame(sapply(df, function(x)(cor(x,df$medv))))
corr_target
```

```
##         sapply.df..function.x...cor.x..df.medv...
## crim                                    -0.3883046
## zn                                       0.3604453
## indus                                   -0.4837252
## chas                                     0.1752602
## nox                                     -0.4273208
## rm                                       0.6953599
## age                                     -0.3769546
## dis                                      0.2499287
## rad                                     -0.3816262
## tax                                     -0.4685359
## ptratio                                 -0.5077867
## black                                    0.3334608
## lstat                                   -0.7376627
## medv                                     1.0000000
```

```
class(corr_target)
```

```
## [1] "data.frame"
```

```
model2 = lm(medv~.,data = df_tr)
summary(model2)
```

```
##
## Call:
## lm(formula = medv ~ ., data = df_tr)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.1704  -2.8897  -0.5358   1.9834  25.7662
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  33.895482   5.676948   5.971 5.32e-09 ***
## crim         -0.111814   0.033121  -3.376  0.00081 ***
## zn            0.044582   0.015167   2.939  0.00349 **
## indus         0.019804   0.069910   0.283  0.77711
## chas          1.818172   0.948817   1.916  0.05606 .
## nox         -16.831857   4.131481  -4.074 5.60e-05 ***
## rm            4.278180   0.449367   9.520  < 2e-16 ***
## age          -0.005923   0.014307  -0.414  0.67911
## dis          -1.433871   0.215001  -6.669 8.83e-11 ***
## rad           0.329867   0.080037   4.121 4.60e-05 ***
## tax          -0.014838   0.004509  -3.291  0.00109 **
## ptratio      -0.976571   0.147682  -6.613 1.25e-10 ***
## black         0.009111   0.002863   3.182  0.00158 **
## lstat        -0.454922   0.055251  -8.234 2.76e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.675 on 390 degrees of freedom
## Multiple R-squared:  0.753,  Adjusted R-squared:  0.7448
## F-statistic: 91.45 on 13 and 390 DF,  p-value: < 2.2e-16
```

```
p2 = predict(model2,df_test)
r_sq=cor(df_test$medv,p2)**2
r_sq
```

```
## [1] 0.678494
```

```
ac_pred = data.frame('Actual'=df_test$medv,'Predicted'=p2)
min_max_accuracy=mean(apply(ac_pred, 1, min)/apply(ac_pred, 1, max))
min_max_accuracy
```

```
## [1] 0.8672125
```

```
mae(df_test$medv,p2)
```

```
## [1] 3.145832
```

```
rmse(df_test$medv,p2)
```

```
## [1] 5.100412
```

Name : Sumon Singh

Assignment : 7

Course : M.sc Computer Science With Specialization in Data Science

Paper : PSDS103

Import Libraries

In [1]:
```
import pandas as pd
import seaborn as sns
import numpy as np
from mlxtend.evaluate import bias_variance_decomp
import matplotlib.pyplot as plt
import pylab
print("Setup done.....")
```

```
Setup done.....
```

Import Dataset

In [2]:
```
df = pd.read_csv('/home/sumon/Desktop/Datasets/Admission_Predict.csv')
```

EDA Of Dataset

In [3]:
```
df.head()
```

Out[3]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

In [4]:
```
df.dtypes
```

Out[4]:
```
Serial No.             int64
GRE Score              int64
TOEFL Score            int64
University Rating      int64
SOP                  float64
LOR                  float64
CGPA                 float64
Research               int64
Chance of Admit      float64
dtype: object
```

In [5]:
```
df.shape
```

Out[5]:
```
(400, 9)
```

In [6]:
```
df.drop('Serial No.',axis=1,inplace=True)
```

In [7]:
```
df.columns
```

Out[7]:
```
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA',
       'Research', 'Chance of Admit '],
      dtype='object')
```

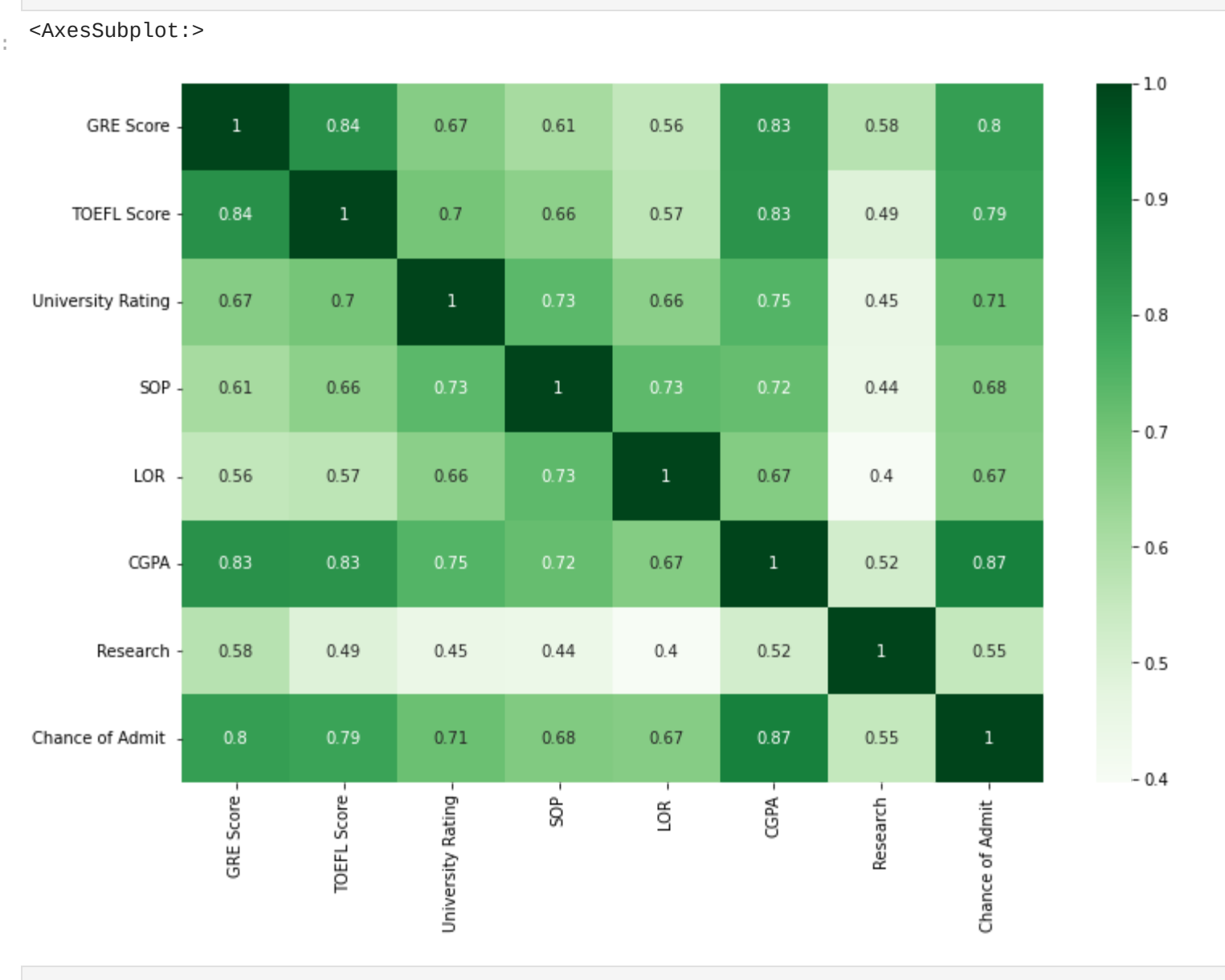
In [8]:
```
df.corr()
```

Out[8]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
GRE Score	1.000000	0.835977	0.668976	0.612831	0.557555	0.833060	0.580391	0.802610
TOEFL Score	0.835977	1.000000	0.695590	0.657981	0.567721	0.828417	0.489858	0.791594
University Rating	0.668976	0.695590	1.000000	0.734523	0.660123	0.746479	0.447783	0.711250
SOP	0.612831	0.657981	0.734523	1.000000	0.729593	0.718144	0.444029	0.675732
LOR	0.557555	0.567721	0.660123	0.729593	1.000000	0.670211	0.396859	0.669889
CGPA	0.833060	0.828417	0.746479	0.718144	0.670211	1.000000	0.521654	0.873289
Research	0.580391	0.489858	0.447783	0.444029	0.396859	0.521654	1.000000	0.553202
Chance of Admit	0.802610	0.791594	0.711250	0.675732	0.669889	0.873289	0.553202	1.000000

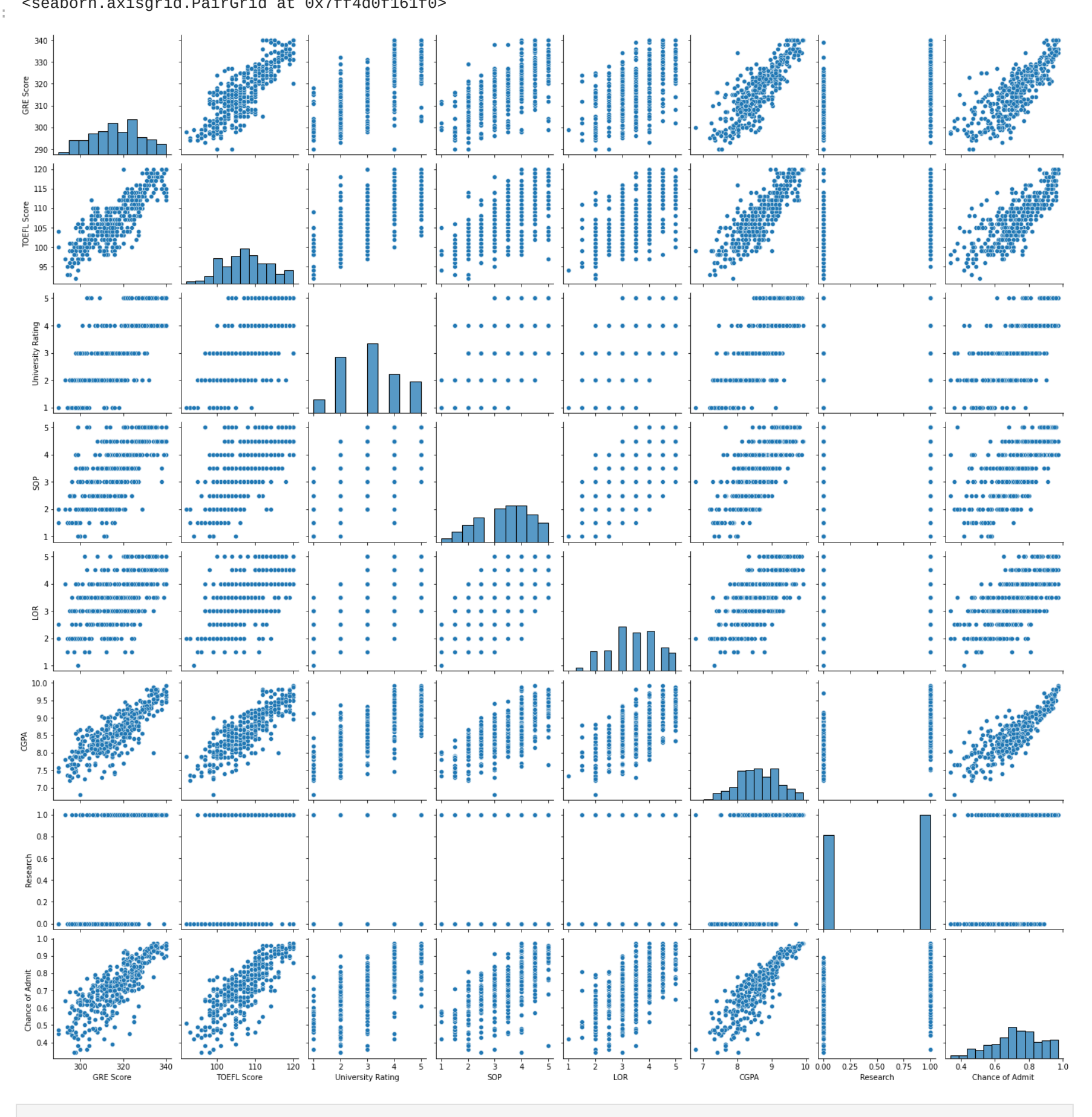
In [9]:
```
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(),annot=True,cmap='Greens')
```

Out[9]:
```
<AxesSubplot:>
```



In [10]:
```
sns.pairplot(df)
```

Out[10]:
```
<seaborn.axisgrid.PairGrid at 0x7ff4d0f161f0>
```



In [11]:
```
df.isnull().sum()
```

Out[11]:
```
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

In [12]:
```
df.skew()
```

Out[12]:
```
GRE Score           -0.062893
TOEFL Score          0.057216
University Rating    0.171260
SOP                 -0.275761
LOR                 -0.106991
CGPA                -0.065991
Research            -0.191582
Chance of Admit     -0.353448
dtype: float64
```

In [13]:
```
df.kurtosis()
```

Out[13]:
```
GRE Score           -0.700489
TOEFL Score         -0.578778
University Rating   -0.796287
SOP                 -0.675610
LOR                 -0.662484
CGPA                -0.458476
Research            -1.973187
Chance of Admit     -0.389126
dtype: float64
```

In [14]:
```
df.std()
```

Out[14]:
```
GRE Score            11.473646
TOEFL Score           6.069514
University Rating     1.143728
SOP                   1.006869
LOR                   0.898478
CGPA                  0.596317
Research              0.498362
Chance of Admit       0.142609
dtype: float64
```

Import libraries for machine learning

In [15]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score,mean_absolute_error,accuracy_score,mean_squared_error
from scipy.stats import skew,kurtosis,probplot
```

Simple Linear regression

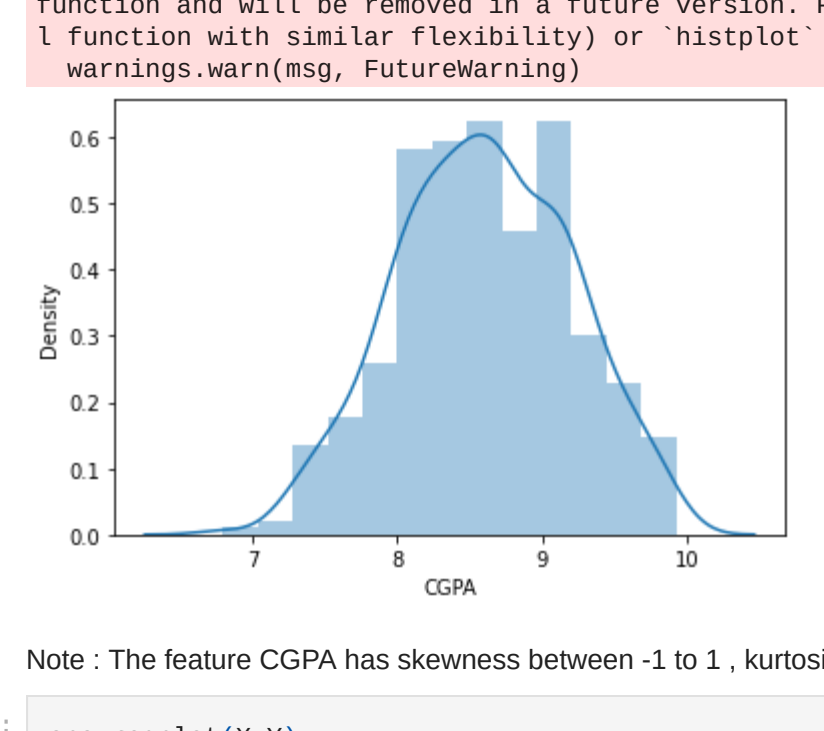
Considering the feature which is most strongly related to the target variable

Note : CGPA has the most strong correlation with Chance of Admit.

In [16]:
```
X = df['CGPA']
Y = df['Chance of Admit ']
```

In [17]:
```
sns.distplot(X)
print('skewness:',skew(X))
print('Kurtosis:',kurtosis(X))
```

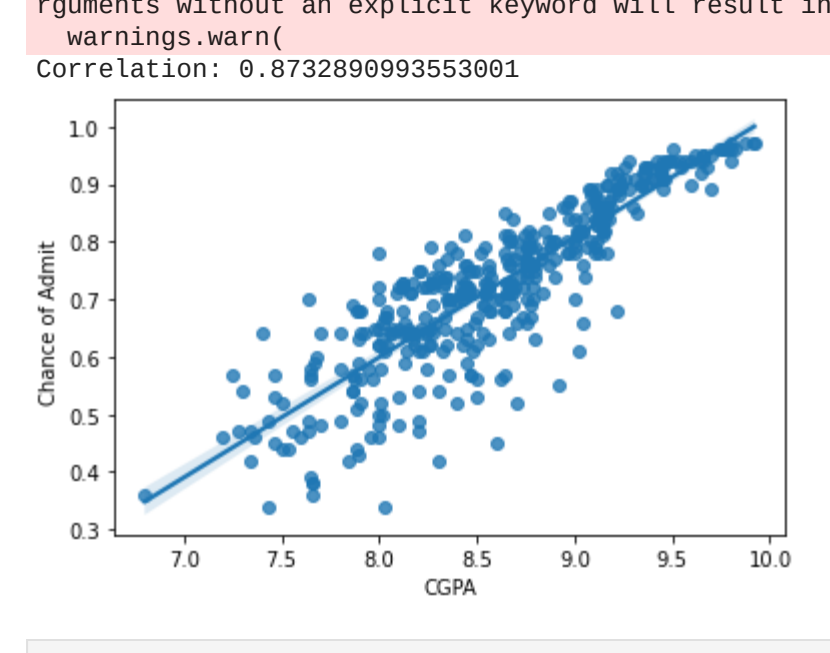
```
skewness: -0.06574282101214378
Kurtosis: -0.4677272965380497
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



Note : The feature CGPA has skewness between -1 to 1 , kurtosis less than 3 so no need to transform the data

In [18]:
```
sns.regplot(X,Y)
print('Correlation:',X.corr(Y))
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
Correlation: 0.8732890993553001
```



In [19]:
```
lm = LinearRegression()
```

In [20]:
```
x_tr,x_test,y_tr,y_test = train_test_split(X,Y,test_size=0.2,random_state=100)
```

In [21]:
```
x_tr=pd.DataFrame(x_tr)
y_tr=pd.DataFrame(y_tr)
x_test=pd.DataFrame(x_test)
y_test=pd.DataFrame(y_test)
```

In [22]:
```
lm.fit(x_tr,y_tr)
```

Out[22]:
```
LinearRegression()
```

In [23]:
```
pred = lm.predict(x_test)
```

In [24]:
```
mean_absolute_error(y_test,pred)
```

Out[24]:
```
0.05118390932402178
```

In [25]:
```
r2_score(y_test,pred)
```

Out[25]:
```
0.735533814457832
```

In [26]:
```
resid = np.array(y_test) - np.array(pred)
```

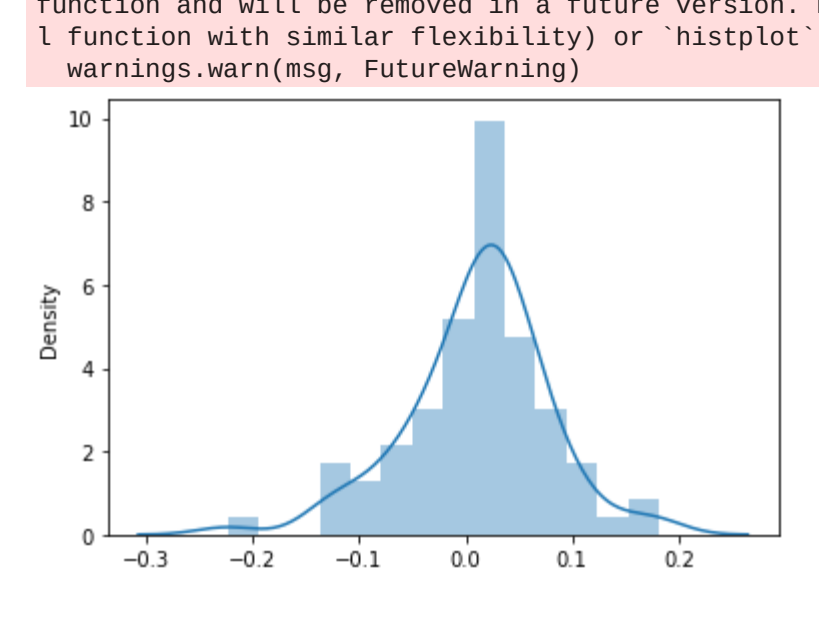
In [27]:
```
resid.shape=(80,)
```

In [28]:
```
resid.shape
```

Out[28]:
```
(80,)
```

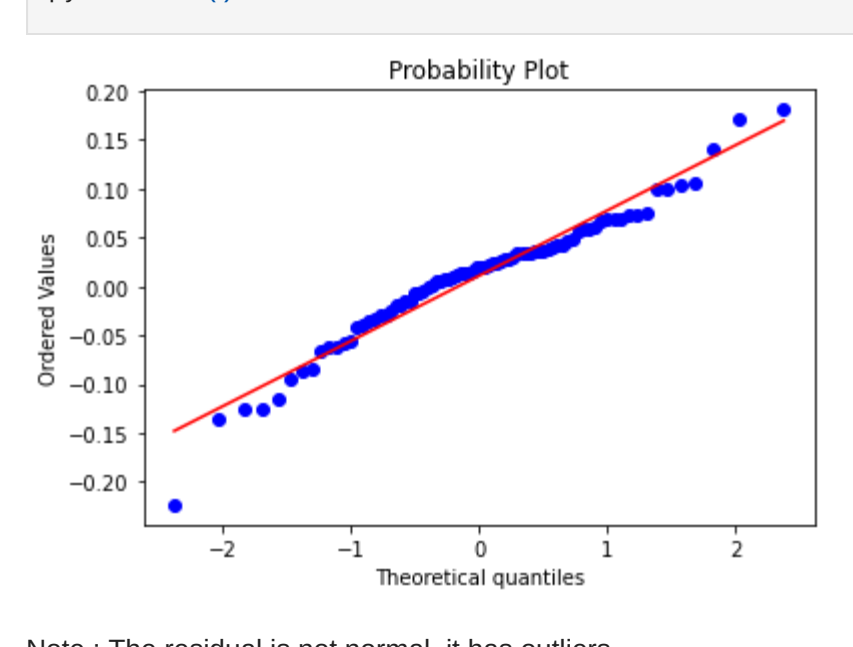
In [29]:
```
sns.distplot(resid)
print('Skewness of residual:',skew(resid))
print('Kurtosis of residual:',kurtosis(resid))
```

```
Skewness of residual: -0.5051010570868172
Kurtosis of residual: 1.496551594859115
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



Q-Q Plot

In [30]:
```
import scipy.stats as s
s.probplot(resid,dist="norm",plot=pylab)
pylab.show()
```



Note : The residual is not normal ,it has outliers .

Shapiro Wilk Test

In [31]:
```
s.shapiro(resid)
```

Out[31]:
```
ShapiroResult(statistic=0.9638625383377075, pvalue=0.023268008604645 73)
```

Conclusion : As the pvalue is less than 0.05 so we reject the null hypothesis of Shapiro Wilk test, the residual is not normal

Durbin Watson Test

In [32]:
```
from statsmodels.stats.stattools import durbin_watson
dbtest = durbin_watson(resid)
dbtest
```

Out[32]:
```
2.4080109986588987
```

Conclusion :As durbin watson test has result between 1.5 to 2.5 so we accept null hypothesis

Multiple Linear regression - Model 1

Considering top 3 features which are most strongly correlated with the target variable

Note : CGPA,GRE Score,TOEFL Score are top3 most correlated with Chance of Admit

In [33]:
```
x_tr,x_test,y_tr,y_test = train_test_split(df[["CGPA","GRE Score","TOEFL Score"]],Y,test_size=0.2,random_state=
```

In [34]:
```
lm.fit(x_tr,y_tr)
```

Out[34]:
```
LinearRegression()
```

In [35]:
```
pred = lm.predict(x_test)
```

In [36]:
```
mse,bias,variance = bias_variance_decomp(lm,x_tr.values,y_tr.values,
                                         x_test.values,y_test.values,
                                         loss='mse',random_seed=1,num_rounds=200)
```

In [37]:
```
print("mse",mse)
print("bias",bias)
print("variance",variance)
```

```
mse 0.004073185895253251
bias 0.004017325511139573
variance 5.586038411367401e-05
```

In [38]:
```
mean_absolute_error(y_test,pred)
```

Out[38]:
```
0.04767985028995394
```

In [39]:
```
r2_score(y_test,pred)
```

Out[39]:
```
0.7665382846218312
```

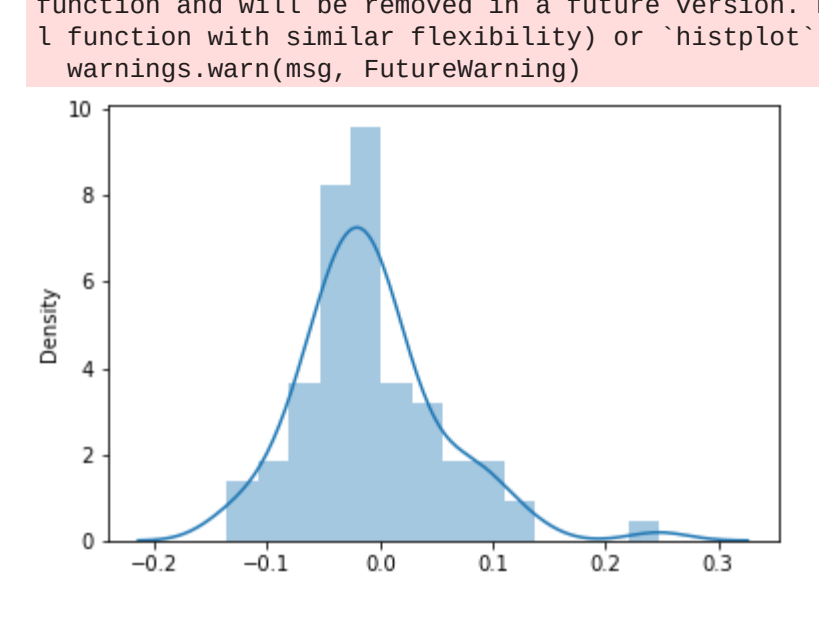
In [40]:
```
resid = np.array(pred) - np.array(y_test)
```

In [41]:
```
resid.shape=(np.product(resid.shape),)
resid.shape
```

Out[41]:
```
(80,)
```

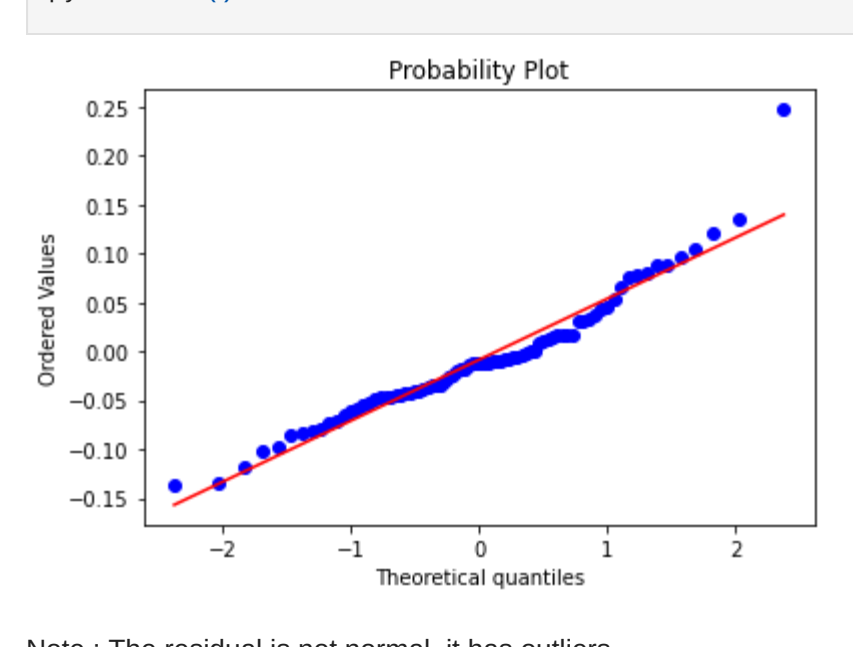
In [42]:
```
sns.distplot(resid)
print('Skewness of residual:',skew(resid))
print('Kurtosis of residual:',kurtosis(resid))
```

```
Skewness of residual: 0.9761698760219699
Kurtosis of residual: 2.393562660444374
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



Q-Q Plot

In [43]:
```
import scipy.stats as s
s.probplot(resid,dist="norm",plot=pylab)
pylab.show()
```



Note : The residual is not normal ,it has outliers .

Shapiro Wilk Test

In [44]:
```
s.shapiro(resid)
```

Out[44]:
```
ShapiroResult(statistic=0.9450209140777588, pvalue=0.0018075279658660293)
```

Conclusion : As the pvalue is less than 0.05 so we reject the null hypothesis of Shapiro Wilk test , The residual is not normal

Durbin Watson Test

In [45]:
```
dbtest = durbin_watson(resid)
dbtest
```

Out[45]:
```
2.3563288245524063
```

Conclusion :As durbin watson test has result between 1.5 to 2.5 so we accept null hypothesis

Multiple Linear regression - Model 2

Considering all the features for target variable

In [46]:
```
X = df.drop('Chance of Admit ',axis=1)
```

In [47]:
```
x_tr,x_test,y_tr,y_test = train_test_split(X,Y,test_size=0.2,random_state=100)
```

In [48]:
```
lm.fit(x_tr,y_tr)
```

Out[48]:
```
LinearRegression()
```

In [49]:
```
pred = lm.predict(x_test)
```

In [50]:
```
mse,bias,variance = bias_variance_decomp(lm,x_tr.values,y_tr.values,
                                         x_test.values,y_test.values,
                                         loss='mse',random_seed=1,num_rounds=200)
```

In [51]:
```
print("mse",mse)
print("bias",bias)
print("variance",variance)
```

```
mse 0.0039027813078997336
bias 0.0037947094685047964
variance 0.00010807183939493335
```

In [52]:
```
mean_absolute_error(y_test,pred)
```

Out[52]:
```
0.047833906161779606
```

In [53]:
```
r2_score(y_test,pred)
```

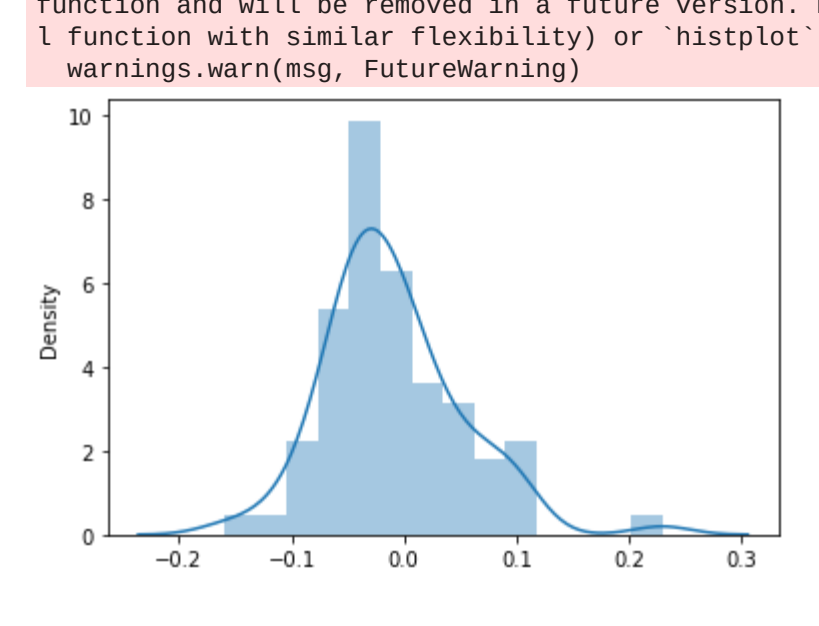
Out[53]:
```
0.7792013613144768
```

In [54]:
```
resid = np.array(pred) - np.array(y_test)
```

In [55]:
```
resid.shape=(np.product(resid.shape),)
```

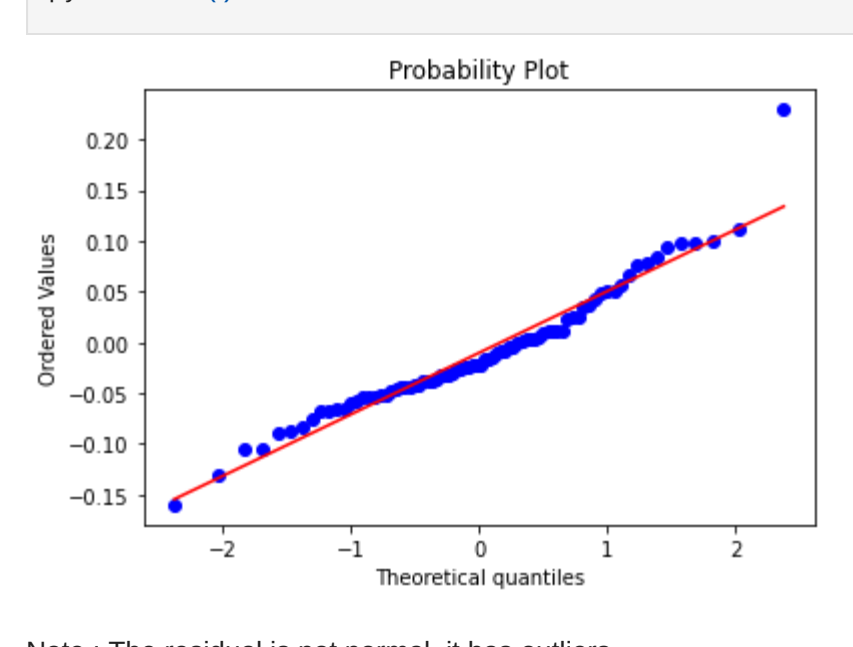
In [56]:
```
sns.distplot(resid)
print('Skewness of residual:',skew(resid))
print('Kurtosis of residual:',kurtosis(resid))
```

```
Skewness of residual: 0.8164699722515354
Kurtosis of residual: 1.9910984096913654
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



Q-Q Plot

In [57]:
```
import scipy.stats as s
s.probplot(resid,dist="norm",plot=pylab)
pylab.show()
```



Note : The residual is not normal ,it has outliers .

Shapiro Wilk Test

In [58]:
```
s.shapiro(resid)
```

Out[58]:
```
ShapiroResult(statistic=0.9549047350883484, pvalue=0.006648587062954903)
```

Conclusion : As the pvalue is less than 0.05 so we reject the null hypothesis of Shapiro Wilk test , there is no normality in residual

Durbin Watson Test

In [59]:
```
dbtest = durbin_watson(resid)
dbtest
```

Out[59]:
```
2.416387363652009
```

Conclusion :As durbin watson test has result between 1.5 to 2.5 so we accept null hypothesis

OLS Model

In [60]:
```
from statsmodels.formula.api import ols
```

In [61]:
```
df_new = df.copy()
```

In [62]:
```
df_new.columns
```

Out[62]:
```
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA',
       'Research', 'Chance of Admit '],
      dtype='object')
```

In [63]:
```
df_new.rename({'Chance of Admit ':'Chance_of_Admit','University Rating':'University_Rating','GRE Score':'GRE_Sc
```

In [64]:
```
df_new
```

Out[64]:

	GRE_Score	TOEFL_Score	University_Rating	SOP	LOR	CGPA	Research	Chance_of_Admit
0	337	118	4	4.5	4.5	9.65	1	0.92
1	324	107	4	4.0	4.5	8.87	1	0.76
2	316	104	3	3.0	3.5	8.00	1	0.72
3	322	110	3	3.5	2.5	8.67	1	0.80
4	314	103	2	2.0	3.0	8.21	0	0.65
...	...	...	...	...	...	...	...	...
395	324	110	3	3.5	3.5	9.04	1	0.82
396	325	107	3	3.0	3.5	9.11	1	0.84
397	330	116	4	5.0	4.5	9.45	1	0.91
398	312	103	3	3.5	4.0	8.78	0	0.67
399	333	117	4	5.0	4.0	9.66	1	0.95

400 rows × 8 columns

In [65]:
```
df_train,df_test = train_test_split(df_new,test_size=0.2,random_state=100)
```

In [66]:
```
ols_model = ols("Chance_of_Admit~GRE_Score+TOEFL_Score+University_Rating+SOP+LOR+CGPA+Research",data=df_train).
```

In [67]:
```
ols_model.summary()
```

Out[67]:

OLS Regression Results

Dep. Variable:	Chance_of_Admit	R-squared:	0.807
Model:	OLS	Adj. R-squared:	0.803
Method:	Least Squares	F-statistic:	186.7
Date:	Tue, 21 Dec 2021	Prob (F-statistic):	1.56e-107
Time:	12:19:51	Log-Likelihood:	427.08
No. Observations:	320	AIC:	-838.2
Df Residuals:	312	BIC:	-808.0
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-1.2797	0.140	-9.160	0.000	-1.555	-1.005
GRE_Score	0.0015	0.001	2.157	0.032	0.000	0.003
TOEFL_Score	0.0034	0.001	2.716	0.007	0.001	0.006
University_Rating	0.0049	0.005	0.919	0.359	-0.006	0.015
SOP	-0.0072	0.006	-1.148	0.252	-0.020	0.005
LOR	0.0251	0.006	3.875	0.000	0.012	0.038
CGPA	0.1265	0.014	9.194	0.000	0.099	0.154
Research	0.0245	0.009	2.709	0.007	0.007	0.042

Omnibus:	69.790	Durbin-Watson:	2.071
Prob(Omnibus):	0.000	Jarque-Bera (JB):	136.189
Skew:	-1.147	Prob(JB):	2.67e-30
Kurtosis:	5.226	Cond. No.	1.30e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.3e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

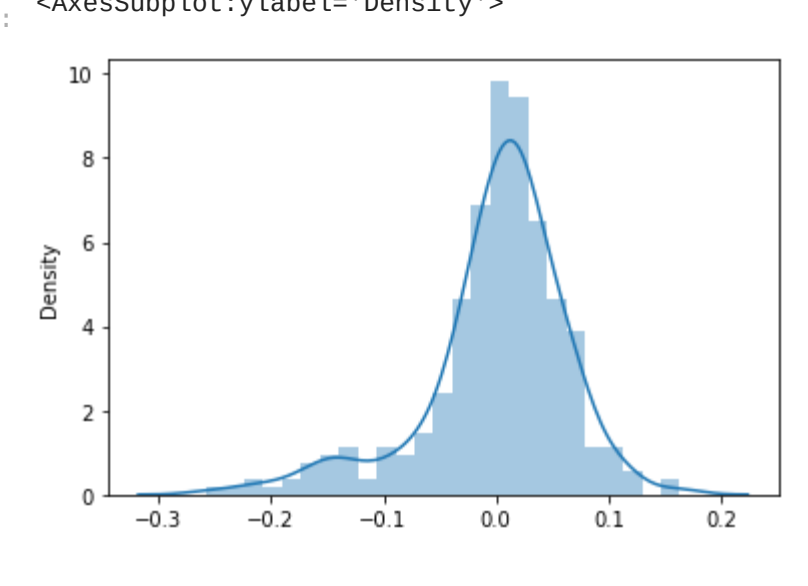
In [68]:
```
fitted = ols_model.fittedvalues
```

In [69]:
```
resid = ols_model.resid
```

In [70]:
```
sns.distplot(resid)
```

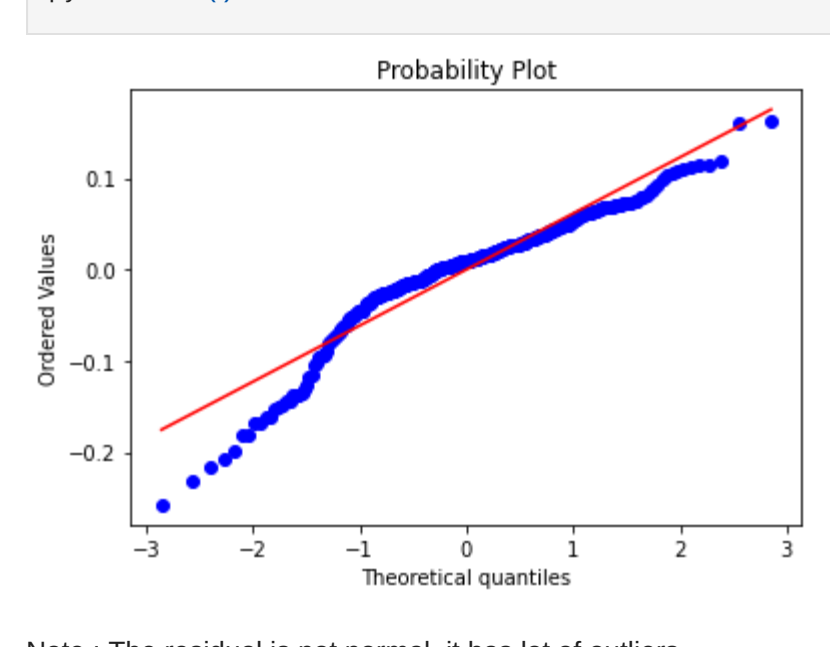
```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[70]:
```
<AxesSubplot:ylabel='Density'>
```



Q-Q Plot

In [71]:
```
import scipy.stats as s
s.probplot(resid,dist="norm",plot=pylab)
pylab.show()
```



Note : The residual is not normal ,it has lot of outliers .

Shapiro Wilk Test

In [72]:
```
s.shapiro(resid)
```

Out[72]:
```
ShapiroResult(statistic=0.9178267121315002, pvalue=3.0958974533285e-12)
```

Conclusion : As the pvalue is less than 0.05 so we reject the null hypothesis of Shapiro Wilk test,there is no normality in residual

Durbin Watson Test

In [73]:

```
dbtest = durbin_watson(resid)
dbtest
```

Out[73]:

```
2.0709355903215134
```

Conclusion :As durbin watson test has result between 1.5 to 2.5 so we accept null hypothesis

Breuschpagan Test

In [74]:

```
import statsmodels.stats.api as sm
sm.het_breuschpagan(ols_model.resid,ols_model.model.exog)[1]
```

Out[74]:

```
0.0016143997762689362
```

As the p-value is less than 0.05 we reject the null hypothesis for breuschpagan

Time_Series (Assignment-7)

Sumon Singh

2022-05-15

Load Library

```
library(timeSeries)
```

```
## Loading required package: timeDate
```

```
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

Read Dataset

```
df = read.csv('/home/sumon/Desktop/Python-Codes/Indrani-mam/Assignment-7/daily-total-female-births.csv')
head(df)
```

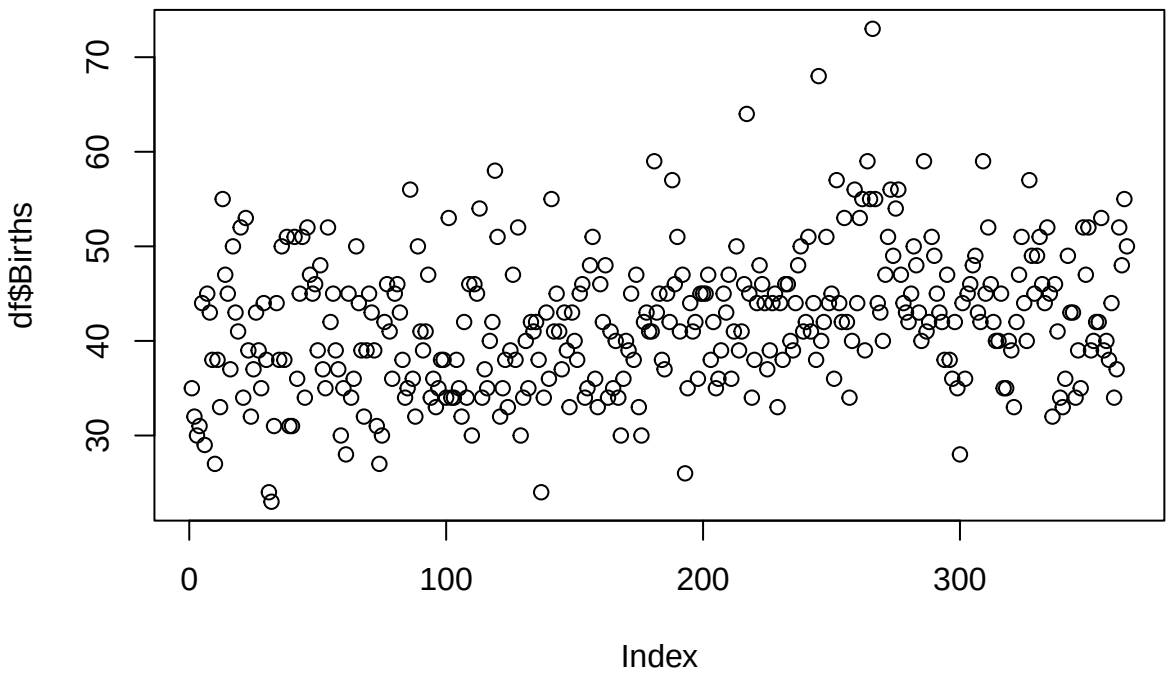
```
##         Date Births
## 1 1959-01-01     35
## 2 1959-01-02     32
## 3 1959-01-03     30
## 4 1959-01-04     31
## 5 1959-01-05     44
## 6 1959-01-06     29
```

Summary

```
summary(df)
```

```
##      Date               Births
##  Length:365         Min.   :23.00
##  Class :character   1st Qu.:37.00
##  Mode  :character   Median :42.00
##                     Mean   :41.98
##                     3rd Qu.:46.00
##                     Max.   :73.00
```

Plot



Data Transformation

```
row.names(df)=df$Date
df_mod=df[,-c(1)]
```

Convert Dataset To TimeSeries

```
df_mod = ts(data = df_mod,frequency = 30)
head(df_mod)
```

```
## Time Series:
## Start = c(1, 1)
## End = c(1, 6)
## Frequency = 30
## [1] 35 32 30 31 44 29
```

Print the start, end and frequency

```
start(df_mod)
```

```
## [1] 1 1
```

```
end(df_mod)
```

```
## [1] 13  5
```

```
frequency(df_mod)
```

```
## [1] 30
```

Decompose the dataset

```
decomposed_data = decompose(df_mod,"multiplicative")
decomposed_data
```

```
## $x
## Time Series:
## Start = c(1, 1)
## End = c(13, 5)
## Frequency = 30
##   [1] 35 32 30 31 44 29 45 43 38 27 38 33 55 47 45 37 50 43 41 52 34 53 39 32 37
##  [26] 43 39 35 44 38 24 23 31 44 38 50 38 51 31 31 51 36 45 51 34 52 47 45 46 39
##  [51] 48 37 35 52 42 45 39 37 30 35 28 45 34 36 50 44 39 32 39 45 43 39 31 27 30
##  [76] 42 46 41 36 45 46 43 38 34 35 56 36 32 50 41 39 41 47 34 36 33 35 38 38 34
## [101] 53 34 34 38 35 32 42 34 46 30 46 45 54 34 37 35 40 42 58 51 32 35 38 33 39
## [126] 47 38 52 30 34 40 35 42 41 42 38 24 34 43 36 55 41 45 41 37 43 39 33 43 40
## [151] 38 45 46 34 35 48 51 36 33 46 42 48 34 41 35 40 34 30 36 40 39 45 38 47 33
## [176] 30 42 43 41 41 59 43 45 38 37 45 42 57 46 51 41 47 26 35 44 41 42 36 45 45
## [201] 45 47 38 42 35 36 39 45 43 47 36 41 50 39 41 46 64 45 34 38 44 48 46 44 37
## [226] 39 44 45 33 44 38 46 46 40 39 44 48 50 41 42 51 41 44 38 68 40 42 51 44 45
## [251] 36 57 44 42 53 42 34 40 56 44 53 55 39 59 55 73 55 44 43 40 47 51 56 49 54
## [276] 56 47 44 43 42 45 50 48 43 40 59 41 42 51 49 45 43 42 38 47 38 36 42 35 28
## [301] 44 36 45 46 48 49 43 42 59 45 52 46 42 40 40 45 35 35 40 39 33 42 47 51 44
## [326] 40 57 49 45 49 51 46 44 52 45 32 46 41 34 33 36 49 43 43 34 39 35 52 47 52
## [351] 39 40 42 42 53 39 40 38 44 34 37 52 48 55 50
##
## $seasonal
## Time Series:
## Start = c(1, 1)
## End = c(13, 5)
## Frequency = 30
##   [1] 0.9658815 0.9655662 1.0349163 0.9548504 1.0562436 1.0600917 1.0485235
##   [8] 1.0579387 0.9302591 0.9603238 1.0494971 1.0506181 0.9363863 0.9607381
```

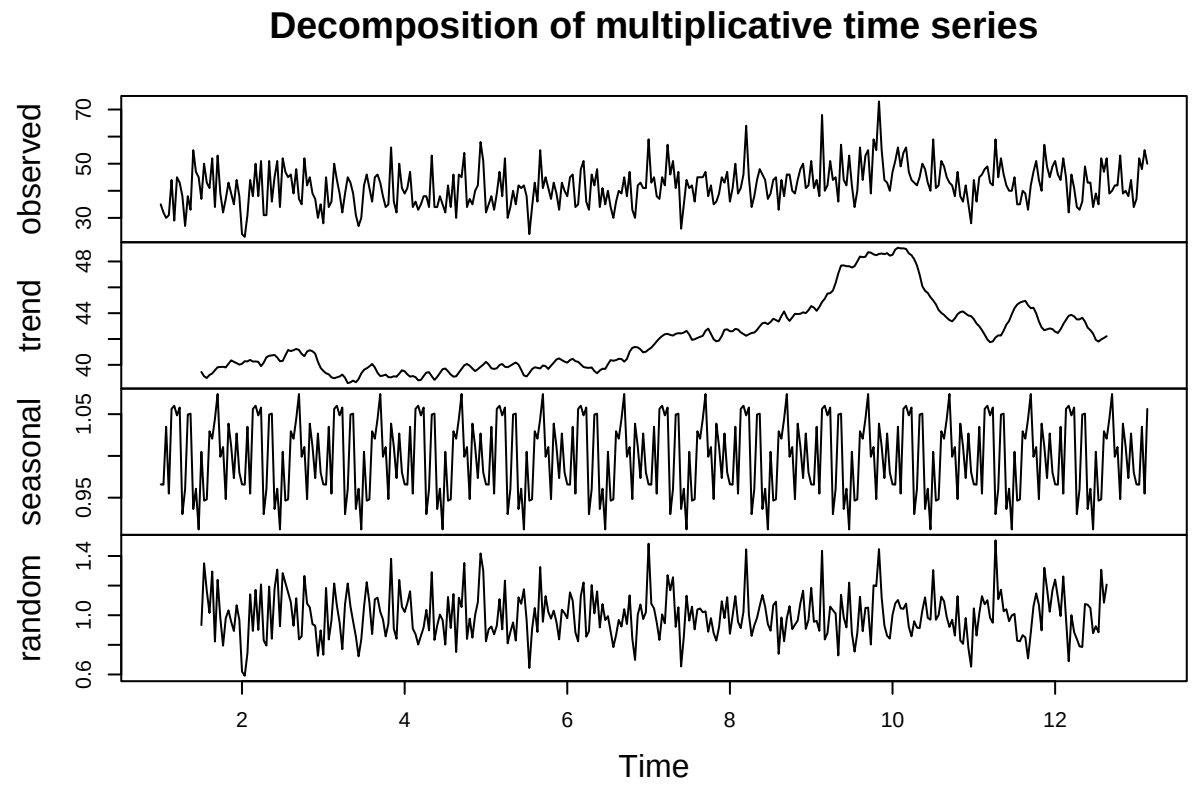
```
##  [15] 0.9119394 1.0049725 0.9464093 0.9479546 1.0295201 1.0206300 1.0443615
##  [22] 1.0741232 0.9989503 1.0106242 0.9482386 1.0386889 1.0112643 0.9734342
##  [29] 1.0268625 0.9801920 0.9658815 0.9655662 1.0349163 0.9548504 1.0562436
##  [36] 1.0600917 1.0485235 1.0579387 0.9302591 0.9603238 1.0494971 1.0506181
##  [43] 0.9363863 0.9607381 0.9119394 1.0049725 0.9464093 0.9479546 1.0295201
##  [50] 1.0206300 1.0443615 1.0741232 0.9989503 1.0106242 0.9482386 1.0386889
##  [57] 1.0112643 0.9734342 1.0268625 0.9801920 0.9658815 0.9655662 1.0349163
##  [64] 0.9548504 1.0562436 1.0600917 1.0485235 1.0579387 0.9302591 0.9603238
##  [71] 1.0494971 1.0506181 0.9363863 0.9607381 0.9119394 1.0049725 0.9464093
##  [78] 0.9479546 1.0295201 1.0206300 1.0443615 1.0741232 0.9989503 1.0106242
##  [85] 0.9482386 1.0386889 1.0112643 0.9734342 1.0268625 0.9801920 0.9658815
##  [92] 0.9655662 1.0349163 0.9548504 1.0562436 1.0600917 1.0485235 1.0579387
##  [99] 0.9302591 0.9603238 1.0494971 1.0506181 0.9363863 0.9607381 0.9119394
## [106] 1.0049725 0.9464093 0.9479546 1.0295201 1.0206300 1.0443615 1.0741232
## [113] 0.9989503 1.0106242 0.9482386 1.0386889 1.0112643 0.9734342 1.0268625
## [120] 0.9801920 0.9658815 0.9655662 1.0349163 0.9548504 1.0562436 1.0600917
## [127] 1.0485235 1.0579387 0.9302591 0.9603238 1.0494971 1.0506181 0.9363863
## [134] 0.9607381 0.9119394 1.0049725 0.9464093 0.9479546 1.0295201 1.0206300
## [141] 1.0443615 1.0741232 0.9989503 1.0106242 0.9482386 1.0386889 1.0112643
## [148] 0.9734342 1.0268625 0.9801920 0.9658815 0.9655662 1.0349163 0.9548504
## [155] 1.0562436 1.0600917 1.0485235 1.0579387 0.9302591 0.9603238 1.0494971
## [162] 1.0506181 0.9363863 0.9607381 0.9119394 1.0049725 0.9464093 0.9479546
## [169] 1.0295201 1.0206300 1.0443615 1.0741232 0.9989503 1.0106242 0.9482386
## [176] 1.0386889 1.0112643 0.9734342 1.0268625 0.9801920 0.9658815 0.9655662
## [183] 1.0349163 0.9548504 1.0562436 1.0600917 1.0485235 1.0579387 0.9302591
## [190] 0.9603238 1.0494971 1.0506181 0.9363863 0.9607381 0.9119394 1.0049725
## [197] 0.9464093 0.9479546 1.0295201 1.0206300 1.0443615 1.0741232 0.9989503
## [204] 1.0106242 0.9482386 1.0386889 1.0112643 0.9734342 1.0268625 0.9801920
## [211] 0.9658815 0.9655662 1.0349163 0.9548504 1.0562436 1.0600917 1.0485235
## [218] 1.0579387 0.9302591 0.9603238 1.0494971 1.0506181 0.9363863 0.9607381
## [225] 0.9119394 1.0049725 0.9464093 0.9479546 1.0295201 1.0206300 1.0443615
## [232] 1.0741232 0.9989503 1.0106242 0.9482386 1.0386889 1.0112643 0.9734342
## [239] 1.0268625 0.9801920 0.9658815 0.9655662 1.0349163 0.9548504 1.0562436
## [246] 1.0600917 1.0485235 1.0579387 0.9302591 0.9603238 1.0494971 1.0506181
## [253] 0.9363863 0.9607381 0.9119394 1.0049725 0.9464093 0.9479546 1.0295201
## [260] 1.0206300 1.0443615 1.0741232 0.9989503 1.0106242 0.9482386 1.0386889
## [267] 1.0112643 0.9734342 1.0268625 0.9801920 0.9658815 0.9655662 1.0349163
## [274] 0.9548504 1.0562436 1.0600917 1.0485235 1.0579387 0.9302591 0.9603238
## [281] 1.0494971 1.0506181 0.9363863 0.9607381 0.9119394 1.0049725 0.9464093
## [288] 0.9479546 1.0295201 1.0206300 1.0443615 1.0741232 0.9989503 1.0106242
## [295] 0.9482386 1.0386889 1.0112643 0.9734342 1.0268625 0.9801920 0.9658815
## [302] 0.9655662 1.0349163 0.9548504 1.0562436 1.0600917 1.0485235 1.0579387
## [309] 0.9302591 0.9603238 1.0494971 1.0506181 0.9363863 0.9607381 0.9119394
## [316] 1.0049725 0.9464093 0.9479546 1.0295201 1.0206300 1.0443615 1.0741232
## [323] 0.9989503 1.0106242 0.9482386 1.0386889 1.0112643 0.9734342 1.0268625
## [330] 0.9801920 0.9658815 0.9655662 1.0349163 0.9548504 1.0562436 1.0600917
## [337] 1.0485235 1.0579387 0.9302591 0.9603238 1.0494971 1.0506181 0.9363863
## [344] 0.9607381 0.9119394 1.0049725 0.9464093 0.9479546 1.0295201 1.0206300
## [351] 1.0443615 1.0741232 0.9989503 1.0106242 0.9482386 1.0386889 1.0112643
## [358] 0.9734342 1.0268625 0.9801920 0.9658815 0.9655662 1.0349163 0.9548504
## [365] 1.0562436
##
## $trend
## Time Series:
```

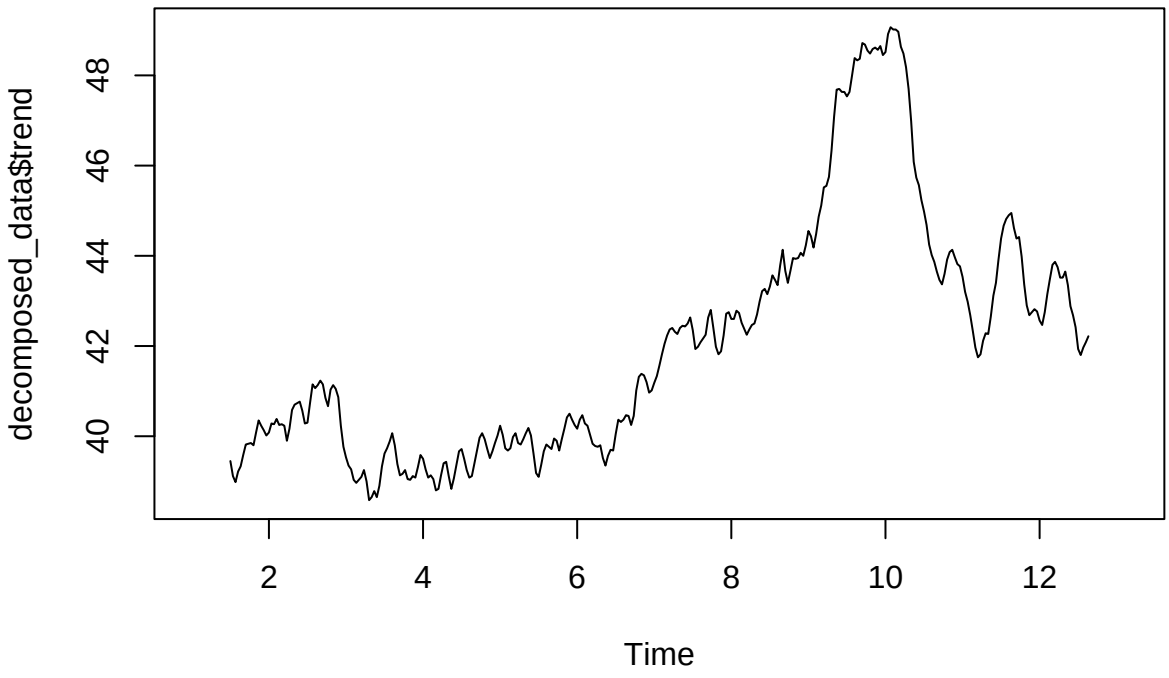
```
## Start = c(1, 1)
## End = c(13, 5)
## Frequency = 30
##   [1]       NA       NA       NA       NA       NA       NA       NA       NA
##   [9]       NA       NA       NA       NA       NA       NA       NA 39.45000
##  [17] 39.11667 38.98333 39.21667 39.33333 39.58333 39.81667 39.83333 39.85000
##  [25] 39.80000 40.08333 40.35000 40.23333 40.13333 40.01667 40.08333 40.28333
##  [33] 40.26667 40.38333 40.25000 40.26667 40.23333 39.90000 40.16667 40.58333
##  [41] 40.70000 40.73333 40.76667 40.56667 40.28333 40.30000 40.73333 41.15000
##  [49] 41.06667 41.13333 41.23333 41.15000 40.85000 40.66667 41.03333 41.13333
##  [57] 41.05000 40.86667 40.23333 39.76667 39.53333 39.35000 39.26667 39.03333
##  [65] 38.96667 39.03333 39.10000 39.25000 39.00000 38.58333 38.65000 38.78333
##  [73] 38.65000 38.90000 39.33333 39.61667 39.73333 39.88333 40.06667 39.80000
##  [81] 39.38333 39.13333 39.16667 39.25000 39.05000 39.03333 39.11667 39.08333
##  [89] 39.31667 39.58333 39.50000 39.26667 39.08333 39.13333 39.05000 38.80000
##  [97] 38.83333 39.13333 39.40000 39.43333 39.11667 38.83333 39.06667 39.36667
## [105] 39.66667 39.71667 39.50000 39.25000 39.08333 39.11667 39.40000 39.68333
## [113] 39.96667 40.06667 39.93333 39.71667 39.51667 39.66667 39.85000 40.01667
## [121] 40.23333 40.03333 39.73333 39.68333 39.73333 39.98333 40.06667 39.85000
## [129] 39.81667 39.93333 40.06667 40.18333 40.01667 39.61667 39.18333 39.10000
## [137] 39.36667 39.66667 39.81667 39.76667 39.71667 39.95000 39.90000 39.68333
## [145] 39.93333 40.16667 40.41667 40.50000 40.36667 40.25000 40.16667 40.36667
## [153] 40.46667 40.28333 40.23333 40.03333 39.83333 39.78333 39.76667 39.80000
## [161] 39.51667 39.35000 39.56667 39.70000 39.68333 40.05000 40.36667 40.31667
## [169] 40.36667 40.46667 40.45000 40.25000 40.45000 41.01667 41.31667 41.38333
## [177] 41.35000 41.20000 40.96667 41.01667 41.18333 41.33333 41.56667 41.81667
## [185] 42.05000 42.23333 42.36667 42.40000 42.31667 42.26667 42.40000 42.45000
## [193] 42.43333 42.50000 42.63333 42.35000 41.93333 41.98333 42.08333 42.16667
## [201] 42.25000 42.63333 42.80000 42.40000 41.98333 41.81667 41.88333 42.23333
## [209] 42.71667 42.75000 42.60000 42.60000 42.78333 42.73333 42.51667 42.38333
## [217] 42.25000 42.36667 42.46667 42.50000 42.70000 42.98333 43.21667 43.26667
## [225] 43.15000 43.31667 43.56667 43.46667 43.35000 43.78333 44.13333 43.66667
## [233] 43.40000 43.66667 43.95000 43.93333 43.95000 44.06667 44.00000 44.23333
## [241] 44.55000 44.43333 44.18333 44.48333 44.86667 45.11667 45.51667 45.55000
## [249] 45.75000 46.33333 47.08333 47.68333 47.70000 47.63333 47.63333 47.53333
## [257] 47.63333 48.00000 48.38333 48.33333 48.36667 48.71667 48.68333 48.55000
## [265] 48.48333 48.58333 48.61667 48.56667 48.65000 48.45000 48.51667 48.91667
## [273] 49.06667 49.01667 49.01667 48.96667 48.63333 48.48333 48.18333 47.70000
## [281] 46.98333 46.08333 45.73333 45.56667 45.23333 44.98333 44.68333 44.25000
## [289] 44.01667 43.86667 43.65000 43.46667 43.36667 43.60000 43.91667 44.08333
## [297] 44.13333 43.96667 43.81667 43.76667 43.53333 43.20000 42.98333 42.68333
## [305] 42.33333 41.96667 41.75000 41.81667 42.11667 42.28333 42.26667 42.65000
## [313] 43.11667 43.40000 43.91667 44.38333 44.66667 44.81667 44.90000 44.95000
## [321] 44.61667 44.38333 44.41667 43.98333 43.36667 42.90000 42.68333 42.75000
## [329] 42.81667 42.76667 42.56667 42.46667 42.75000 43.15000 43.48333 43.80000
## [337] 43.86667 43.75000 43.51667 43.51667 43.65000 43.35000 42.88333 42.68333
## [345] 42.41667 41.93333 41.80000 41.96667 42.08333 42.21667       NA       NA
## [353]       NA       NA       NA       NA       NA       NA       NA       NA
## [361]       NA       NA       NA       NA       NA
##
## $random
## Time Series:
## Start = c(1, 1)
## End = c(13, 5)
```

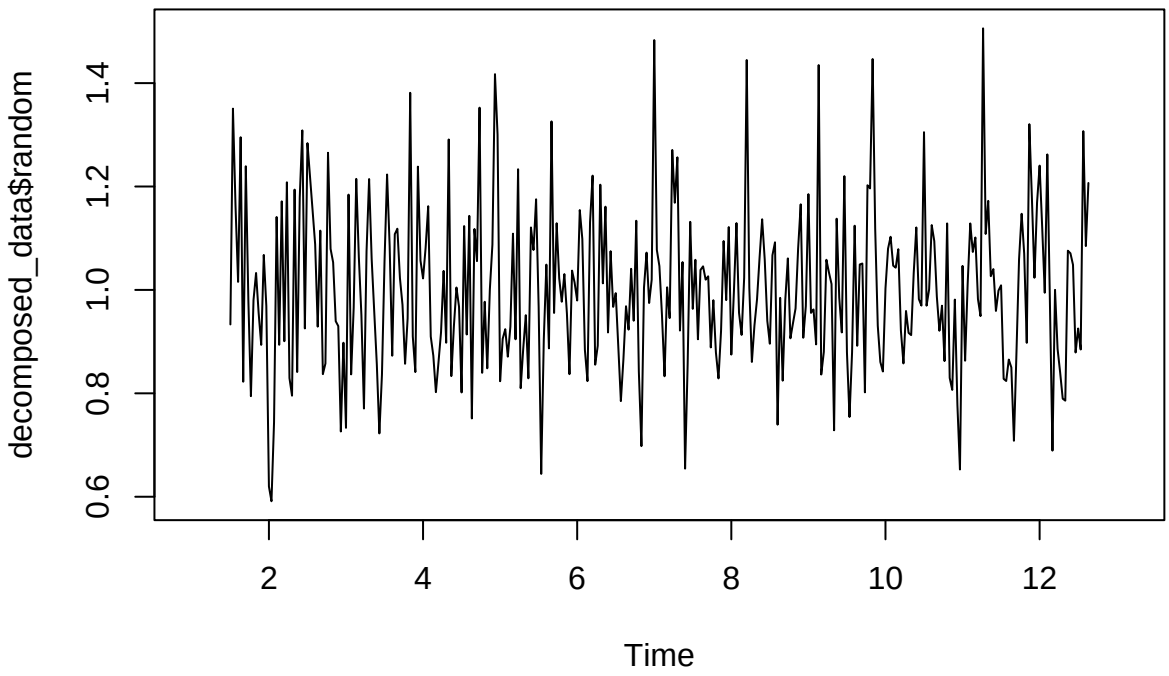
```
## Frequency = 30
##   [1]        NA        NA        NA        NA        NA        NA        NA
##   [8]        NA        NA        NA        NA        NA        NA        NA
##  [15]        NA 0.9332554 1.3506075 1.1635952 1.0154964 1.2953117 0.8224618
##  [22] 1.2392442 0.9801083 0.7945696 0.9803949 1.0328069 0.9557766 0.8936664
##  [29] 1.0676654 0.9687942 0.6199028 0.5913170 0.7438935 1.1410776 0.8938273
##  [36] 1.1713344 0.9007814 1.2081943 0.8296443 0.7954196 1.1939731 0.8412163
##  [43] 1.1788330 1.3085667 0.9255237 1.2839382 1.2191830 1.1535997 1.0880117
##  [50] 0.9289715 1.1146588 0.8371009 0.8576935 1.2652463 1.0794310 1.0532540
##  [57] 0.9394783 0.9300920 0.7261443 0.8979201 0.7332815 1.1843654 0.8366612
##  [64] 0.9658986 1.2148221 1.0633436 0.9512829 0.7706369 1.0749693 1.2144931
##  [71] 1.0600778 0.9571381 0.8565588 0.7224523 0.8363625 1.0549143 1.2232742
##  [78] 1.0844383 0.8727392 1.1077994 1.1183932 1.0229809 0.9712323 0.8571357
##  [85] 0.9452123 1.3812328 0.9100725 0.8411080 1.2384573 1.0567209 1.0222183
##  [92] 1.0813786 1.1619863 0.9099064 0.8728053 0.8023037 0.8595779 0.9178596
##  [99] 1.0367724 0.8978374 1.2910195 0.8333537 0.9294318 1.0047313 0.9675566
## [106] 0.8017205 1.1235003 0.9138012 1.1432242 0.7514344 1.1179201 1.0557237
## [113] 1.3525457 0.8396649 0.9771214 0.8484177 1.0009560 1.0877197 1.4173835
## [120] 1.3002237 0.8234555 0.9054495 0.9241093 0.8709043 0.9292777 1.1088567
## [127] 0.9045285 1.2334300 0.8099392 0.8865957 0.9512519 0.8290433 1.1208651
## [134] 1.0772114 1.1753899 0.9670583 0.6441746 0.9042024 1.0489837 0.8869824
## [141] 1.3259864 0.9554610 1.1290047 1.0223180 0.9771214 1.0306641 0.9542000
## [148] 0.8370518 1.0373690 1.0138715 0.9794764 1.1545362 1.0983864 0.8839306
## [155] 0.8236031 1.1310350 1.2210835 0.8553440 0.8920533 1.2035304 1.0127162
## [162] 1.1610518 0.9176867 1.0749502 0.9671502 0.9938098 0.8899734 0.7849628
## [169] 0.8662531 0.9684880 0.9231988 1.0408605 0.9404186 1.1338296 0.8423082
## [176] 0.6979274 1.0044055 1.0721724 0.9746326 1.0197937 1.4832238 1.0774223
## [183] 1.0460732 0.9516973 0.8330511 1.0051103 0.9454680 1.2707160 1.1685369
## [190] 1.2564768 0.9213757 1.0538414 0.6543516 0.8571841 1.1317159 0.9633326
## [197] 1.0583051 0.9045614 1.0386460 1.0456225 1.0198469 1.0263476 0.8887834
## [204] 0.9801527 0.8791713 0.8288340 0.9207859 1.0945877 0.9802996 1.1216324
## [211] 0.8749214 0.9967636 1.1292501 0.9557900 0.9129786 1.0238098 1.4446915
## [218] 1.0039861 0.8606504 0.9310585 0.9818465 1.0629094 1.1367148 1.0585082
## [225] 0.9402752 0.8958915 1.0671349 1.0921156 0.7394180 0.9846356 0.8244532
## [232] 0.9807396 1.0610216 0.9064007 0.9358109 0.9642131 1.0799849 1.1656098
## [239] 0.9074420 0.9686981 1.1852191 0.9556369 0.9622523 0.8946454 1.4348980
## [246] 0.8363336 0.8800365 1.0583304 1.0338502 1.0113495 0.7285411 1.1377933
## [253] 0.9850976 0.9177689 1.2201098 0.8792185 0.7542042 0.8790857 1.1242358
## [260] 0.8919441 1.0492497 1.0510685 0.8019373 1.2024668 1.1963344 1.4466053
## [267] 1.1186979 0.9306958 0.8607427 0.8422772 1.0029587 1.0797700 1.1027987
## [274] 1.0469284 1.0430039 1.0788077 0.9216917 0.8578270 0.9593292 0.9168815
## [281] 0.9126147 1.0327168 1.1208651 0.9822368 0.9696956 1.3051072 0.9695256
## [288] 1.0012637 1.1254293 1.0944430 0.9871370 0.9209966 0.9695035 0.8623973
## [295] 1.1286281 0.8298960 0.8066239 0.9813392 0.7778868 0.6526846 1.0464221
## [302] 0.8630514 1.0115962 1.1286627 1.0734818 1.1014079 0.9822767 0.9493784
## [309] 1.5058929 1.1082190 1.1722605 1.0265826 1.0402772 0.9593239 0.9987681
## [316] 1.0088774 0.8279526 0.8238363 0.8653242 0.8500933 0.7082164 0.8809987
## [323] 1.0592733 1.1473405 1.0699883 0.8976710 1.3205408 1.1774795 1.0234989
## [330] 1.1689058 1.2404427 1.1218314 0.9945150 1.2620809 0.9797733 0.6891796
## [337] 1.0001037 0.8858196 0.8398841 0.7896608 0.7858452 1.0758757 1.0708407
## [344] 1.0485886 0.8789748 0.9254459 0.8847341 1.3071075 1.0848081 1.2068439
## [351]        NA        NA        NA        NA        NA        NA        NA
## [358]        NA        NA        NA        NA        NA        NA        NA
## [365]        NA
```

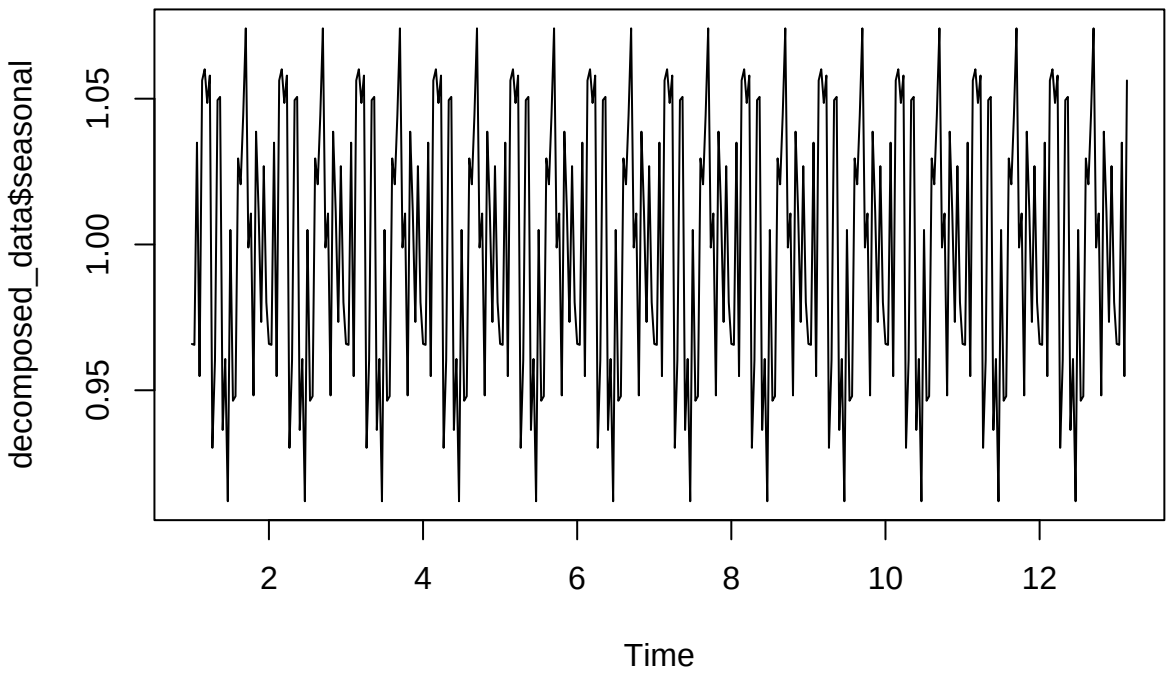
```
##
## $figure
##  [1] 0.9658815 0.9655662 1.0349163 0.9548504 1.0562436 1.0600917 1.0485235
##  [8] 1.0579387 0.9302591 0.9603238 1.0494971 1.0506181 0.9363863 0.9607381
## [15] 0.9119394 1.0049725 0.9464093 0.9479546 1.0295201 1.0206300 1.0443615
## [22] 1.0741232 0.9989503 1.0106242 0.9482386 1.0386889 1.0112643 0.9734342
## [29] 1.0268625 0.9801920
##
## $type
## [1] "multiplicative"
##
## attr(,"class")
## [1] "decomposed.ts"
```

Explore its components





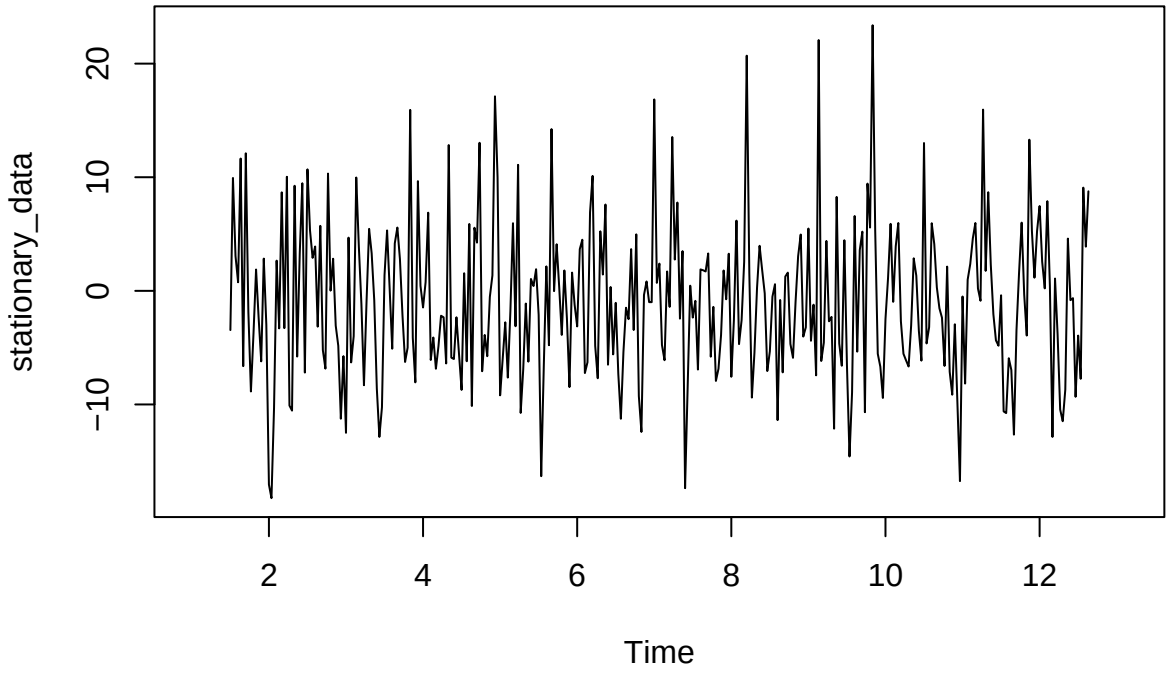

decomposed_data$random
0.6
0.8
1.0
1.2
1.4
2
4
6
8
10
12
Time



Remove the seasonality and trend

```
stationary_data=df_mod-decomposed_data$seasonal-decomposed_data$trend
```

Plot stationary_data



Apply ARIMA

```
f_model=auto.arima(stationary_data)
f_model
```

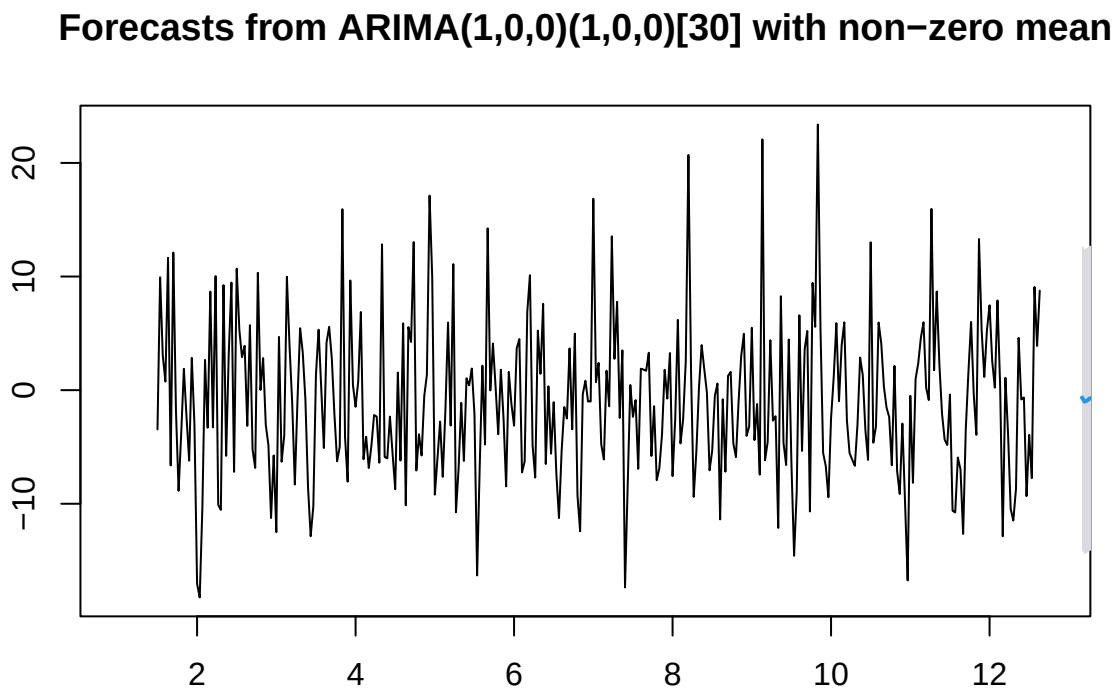
```
## Series: stationary_data
## ARIMA(1,0,0)(1,0,0)[30] with non-zero mean
##
## Coefficients:
##          ar1     sar1     mean
##       0.0965  -0.0264  -0.9689
## s.e.  0.0081   0.0038   0.4005
##
## sigma^2 = 46.44:  log likelihood = -1116.72
## AIC=2241.45   AICc=2241.57   BIC=2256.7
```

Forecast the number of female births for the next 5 months

```
f=forecast(f_model,h=5,level=c(95))
f
```

```
##          Point Forecast     Lo 95    Hi 95
## 13.16667     -0.6546930 -14.07381 12.76443
## 13.20000     -1.0231710 -14.44229 12.39595
## 13.23333     -0.8938857 -14.31300 12.52523
## 13.26667     -0.7184581 -14.13758 12.70066
## 13.30000     -0.6912399 -14.11036 12.72788
```

plot the forecast



Name : Sumon Singh

Assignment : 8

Course : M.sc Computer Science With Specialization in Data Science

Paper : PSDS103

Import libraries

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from scipy.stats import skew
import matplotlib.pyplot as plt
```

Read the boston dataset

In [2]:
```
df_boston = pd.read_csv('/home/sumon/Desktop/Datasets/Boston.csv')
```

In [3]:
```
df_boston.head()
```

Out[3]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

In [4]:
```
df_boston.dtypes
```

Out[4]:
```
Unnamed: 0      int64
crim          float64
zn            float64
indus         float64
chas            int64
nox           float64
rm            float64
age           float64
dis           float64
rad             int64
tax             int64
ptratio       float64
black         float64
lstat         float64
medv          float64
dtype: object
```

In [5]:
```
df_boston.drop('Unnamed: 0',axis=1,inplace=True)
```

In [6]:
```
df_boston.head()
```

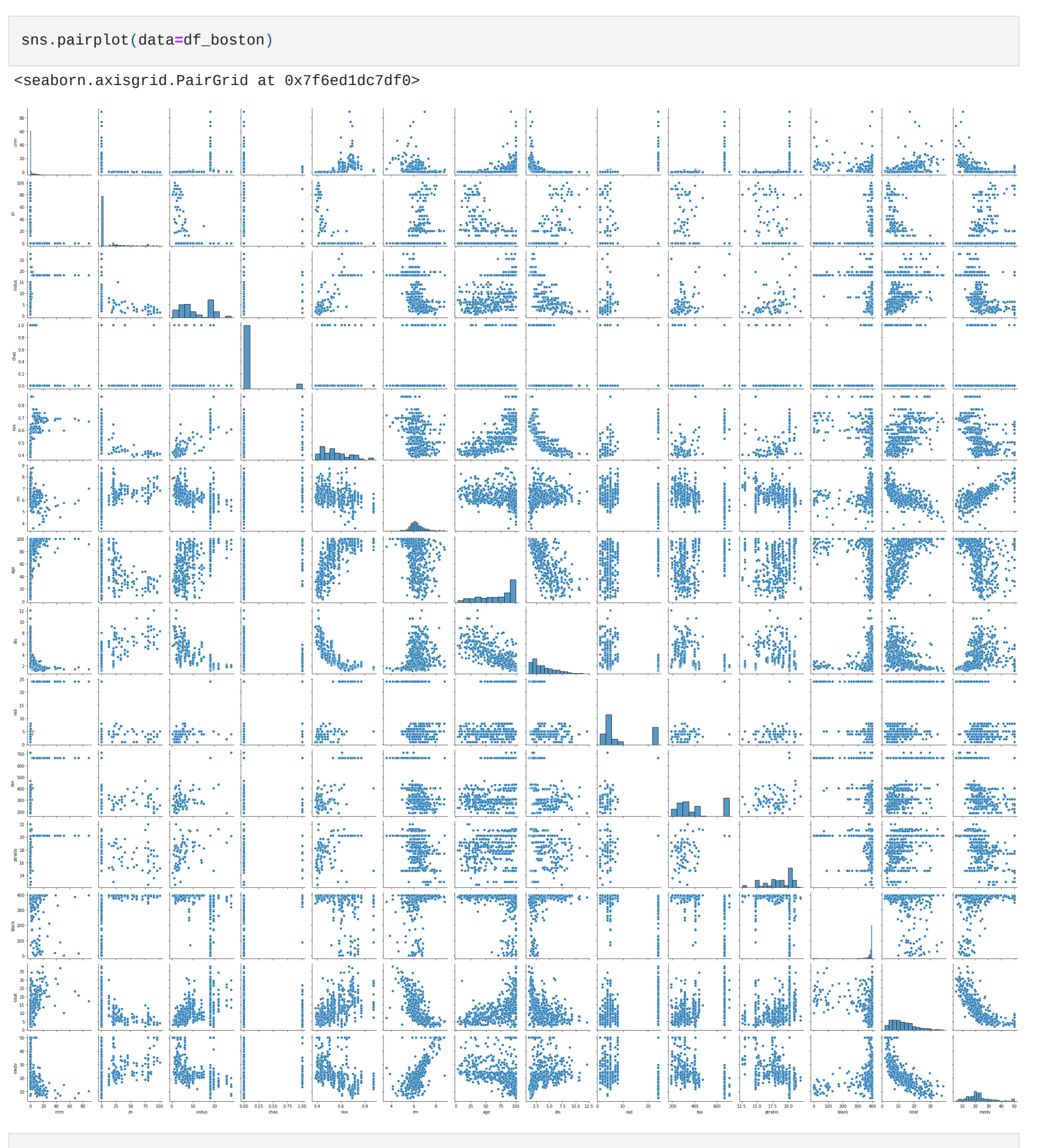
Out[6]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

Check relationship among all the columns of boston dataset

In [7]:
```
sns.pairplot(data=df_boston)
```

Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x7f6ed1dc7df0>
```



In [8]:
```
df_boston.corr()
```

Out[8]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black
crim	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0.379670	0.625505	0.582764	0.289946	-0.385064
zn	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.664408	-0.311948	-0.314563	-0.391679	0.175520
indus	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.708027	0.595129	0.720760	0.383248	-0.356977
chas	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.099176	-0.007368	-0.035587	-0.121515	0.048788
nox	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.769230	0.611441	0.668023	0.188933	-0.380051
rm	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.205246	-0.209847	-0.292048	-0.355501	0.128069
age	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.747881	0.456022	0.506456	0.261515	-0.273534
dis	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.000000	-0.494588	-0.534432	-0.232471	0.291512
rad	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.494588	1.000000	0.910228	0.464741	-0.444413
tax	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.534432	0.910228	1.000000	0.460853	-0.441808
ptratio	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.232471	0.464741	0.460853	1.000000	-0.177383
black	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	0.291512	-0.444413	-0.441808	-0.177383	1.000000
lstat	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.496996	0.488676	0.543993	0.374044	-0.366087
medv	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0.249929	-0.381626	-0.468536	-0.507787	0.333461

In [9]:
```
plt.figure(figsize=(12,5))
sns.heatmap(df_boston.corr(),annot=True)
```

Out[9]:
```
<AxesSubplot:>
```

Machine Learning using Linear Regression

Model 1

Linear regression model with lstat and rm

In [10]:
```
x = df_boston.iloc[:,[5,12]]
```

In [11]:
```
x.head()
```

Out[11]:

	rm	lstat
0	6.575	4.98
1	6.421	9.14
2	7.185	4.03
3	6.998	2.94
4	7.147	5.33

In [12]:
```
y = df_boston.iloc[:,13]
```

In [13]:
```
y.head()
```

Out[13]:
```
0    24.0
1    21.6
2    34.7
3    33.4
4    36.2
Name: medv, dtype: float64
```

In [14]:
```
from sklearn.model_selection import train_test_split
```

In [15]:
```
X_tr,X_test,Y_tr,Y_test = train_test_split(x,y,test_size=0.2,random_state=100)
```

In [16]:
```
X_tr.head()
```

Out[16]:

	rm	lstat
379	6.223	21.78
311	6.122	5.98
157	6.943	4.59
244	5.593	12.50
56	6.383	5.77

In [17]:
```
from sklearn.linear_model import LinearRegression
```

In [18]:
```
modle_lm = LinearRegression().fit(X_tr,Y_tr)
```

In [19]:
```
p_lm = modle_lm.predict(X_test)
```

In [20]:
```
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
```

In [21]:
```
r2_score(Y_test,p_lm)
```

Out[21]:
```
0.6631251961250051
```

In [22]:
```
mean_absolute_error(Y_test,p_lm)
```

Out[22]:
```
4.099161920009365
```

Model 2

Ols

In [23]:
```
df_tr,df_test = train_test_split(df_boston,test_size=0.2,random_state=100)
```

In [24]:
```
from statsmodels.formula.api import ols
```

In [25]:
```
model_ols = ols("medv~crim +zn +indus+chas+nox+rm+age+dis+rad+tax+ptratio+black+lstat",
                data=df_tr).fit()
```

In [26]:
```
model_ols.summary()
```

Out[26]:

OLS Regression Results

Dep. Variable:	medv	R-squared:	0.734
Model:	OLS	Adj. R-squared:	0.725
Method:	Least Squares	F-statistic:	82.67
Date:	Sat, 11 Dec 2021	Prob (F-statistic):	2.38e-103
Time:	22:11:58	Log-Likelihood:	-1194.4
No. Observations:	404	AIC:	2417.
Df Residuals:	390	BIC:	2473.
Df Model:	13		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	36.3338	5.723	6.349	0.000	25.082	47.585
crim	-0.0815	0.044	-1.870	0.062	-0.167	0.004
zn	0.0480	0.015	3.127	0.002	0.018	0.078
indus	-0.0055	0.070	-0.078	0.938	-0.144	0.133
chas	3.0626	0.915	3.345	0.001	1.263	4.862
nox	-16.1369	4.262	-3.786	0.000	-24.516	-7.758
rm	3.6725	0.473	7.765	0.000	2.743	4.602
age	-0.0085	0.015	-0.573	0.567	-0.038	0.021
dis	-1.5174	0.228	-6.660	0.000	-1.965	-1.069
rad	0.2873	0.074	3.888	0.000	0.142	0.433
tax	-0.0121	0.004	-2.893	0.004	-0.020	-0.004
ptratio	-0.9242	0.148	-6.241	0.000	-1.215	-0.633
black	0.0095	0.003	3.006	0.003	0.003	0.016
lstat	-0.4859	0.057	-8.518	0.000	-0.598	-0.374

Omnibus:	134.941	Durbin-Watson:	2.029
Prob(Omnibus):	0.000	Jarque-Bera (JB):	517.166
Skew:	1.452	Prob(JB):	5.00e-113
Kurtosis:	7.722	Cond. No.	1.51e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.51e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

In [27]:
```
p_ols=model_ols.predict(df_test)
```

In [28]:
```
r2_score(df_test["medv"],p_ols)
```

Out[28]:
```
0.7555033086871302
```

In [29]:
```
explained_varience = np.sum((np.array(df_test["medv"])-np.array(p_ols))**2)
total_varience=np.sum((np.array(df_test["medv"])-np.mean(df_test["medv"]))**2)
```

In [30]:
```
r2sq=1-(explained_varience/total_varience)
```

In [31]:
```
r2sq
```

Out[31]:
```
0.7555033086871302
```

In [32]:
```
df_test.shape
```

Out[32]:
```
(102, 14)
```

In [33]:
```
adjr2=1-(1-0.7555033086871302)*(102-1)/(102-13-1)
adjr2
```

Out[33]:
```
0.7193844792886381
```

Model 3

Removing the insignificant variable and create Ols model

In [34]:
```
model_ols2=ols("medv~crim +zn+chas+nox+rm+dis+rad+tax+ptratio+black+lstat",
               data=df_tr).fit()
```

In [35]:
```
model_ols2.summary()
```

Out[35]:

OLS Regression Results

Dep. Variable:	medv	R-squared:	0.734
Model:	OLS	Adj. R-squared:	0.726
Method:	Least Squares	F-statistic:	98.09
Date:	Sat, 11 Dec 2021	Prob (F-statistic):	2.86e-105
Time:	22:11:59	Log-Likelihood:	-1194.6
No. Observations:	404	AIC:	2413.
Df Residuals:	392	BIC:	2461.
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	36.6080	5.683	6.442	0.000	25.435	47.781
crim	-0.0813	0.043	-1.874	0.062	-0.167	0.004
zn	0.0493	0.015	3.264	0.001	0.020	0.079
chas	3.0366	0.908	3.346	0.001	1.252	4.821
nox	-16.8604	3.972	-4.245	0.000	-24.669	-9.052
rm	3.6210	0.461	7.856	0.000	2.715	4.527
dis	-1.4731	0.210	-7.013	0.000	-1.886	-1.060
rad	0.2916	0.070	4.151	0.000	0.154	0.430
tax	-0.0123	0.004	-3.350	0.001	-0.020	-0.005
ptratio	-0.9323	0.146	-6.389	0.000	-1.219	-0.645
black	0.0094	0.003	2.981	0.003	0.003	0.016
lstat	-0.4975	0.053	-9.373	0.000	-0.602	-0.393

Omnibus:	132.820	Durbin-Watson:	2.032
Prob(Omnibus):	0.000	Jarque-Bera (JB):	497.894
Skew:	1.435	Prob(JB):	7.65e-109
Kurtosis:	7.619	Cond. No.	1.47e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.47e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

In [36]:
```
p_ols2 = model_ols2.predict(df_test)
```

In [37]:
```
r2_score(df_test["medv"],p_ols2)
```

Out[37]:
```
0.757481228324034
```

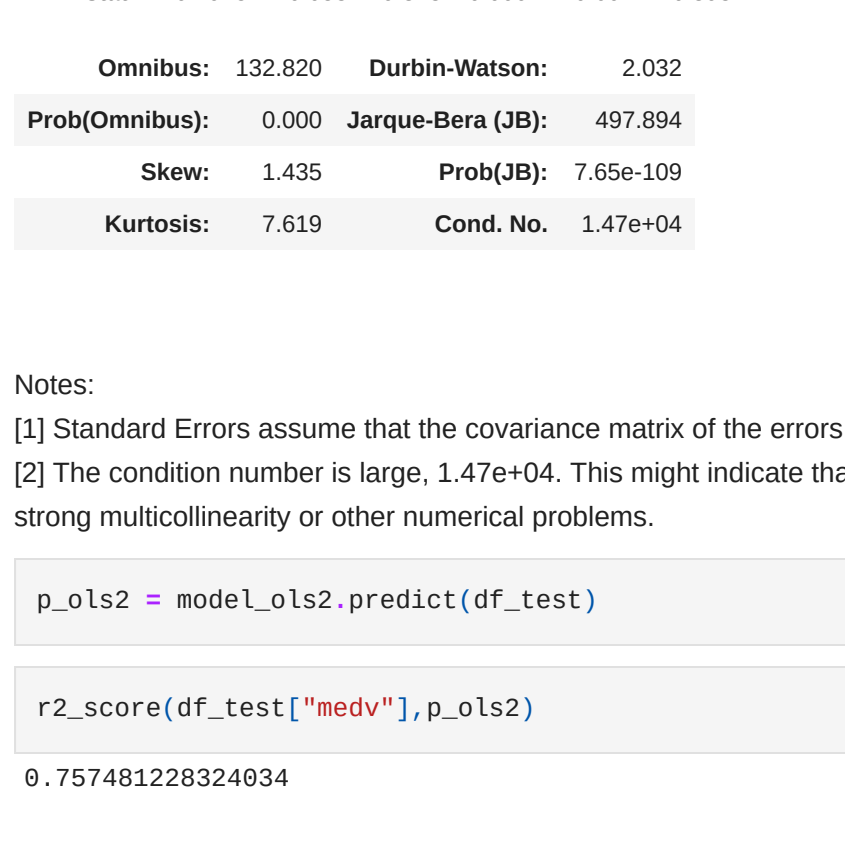
Model 4

Log transformation of 'medv' column

In [38]:
```
sns.distplot(df_boston["medv"])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[38]:
```
<AxesSubplot:xlabel='medv', ylabel='Density'>
```



In [39]:
```
skew(df_boston["medv"])
```

Out[39]:
```
1.104810822864635
```

In [40]:
```
df_boston["log_medv"]=np.log(df_boston["medv"])
```

In [41]:
```
skew(df_boston["log_medv"])
```

Out[41]:
```
-0.3293412745315194
```

In [42]:
```
x = df_boston.iloc[:,0:13]
y = df_boston.iloc[:,14]
```

In [43]:
```
x_tr,x_test,y_tr,y_test = train_test_split(x,y,test_size=0.2,random_state=100)
```

In [44]:
```
model_log = LinearRegression().fit(x_tr,y_tr)
```

In [45]:
```
p_log = model_log.predict(x_test)
```

In [46]:
```
r2_score(p_log,y_test)
```

Out[46]:
```
0.6678274289393569
```

Model 5

Log transformation of skewed columns

In [47]:
```
df_boston = pd.read_csv('/home/sumon/Desktop/Datasets/Boston.csv')
df_boston.drop('Unnamed: 0',axis=1,inplace=True)
```

In [48]:
```
df_boston.skew()
```

Out[48]:
```
crim       5.223149
zn         2.225666
indus      0.295022
chas       3.405904
nox        0.729308
rm         0.403612
age       -0.598963
dis        1.011781
rad        1.004815
tax        0.669956
ptratio   -0.802325
black     -2.890374
lstat      0.906460
medv       1.108098
dtype: float64
```

In [49]:
```
df_boston.kurtosis()
```

Out[49]:
```
crim       37.130509
zn          4.031510
indus      -1.233540
chas        9.638264
nox        -0.064667
rm          1.891500
age        -0.967716
dis         0.487941
rad        -0.867232
tax        -1.142408
ptratio    -0.285091
black       7.226818
lstat       0.493240
medv        1.495197
dtype: float64
```

In [50]:
```
df_kurtosis = df_boston.kurtosis()
df_kurtosis[(df_kurtosis>3)].index
```

Out[50]:
```
Index(['crim', 'zn', 'chas', 'black'], dtype='object')
```

In [51]:
```
df_skew = df_boston.skew()
```

In [52]:
```
col = df_skew[((df_skew>1.0) | (df_skew<-1.0))].index
col
```

Out[52]:
```
Index(['crim', 'zn', 'chas', 'dis', 'rad', 'black', 'medv'], dtype='object')
```

In [53]:
```
df_skewed = df_boston[col]
```

In [54]:
```
df_skewed.head()
```

Out[54]:

	crim	zn	chas	dis	rad	black	medv
0	0.00632	18.0	0	4.0900	1	396.90	24.0
1	0.02731	0.0	0	4.9671	2	396.90	21.6
2	0.02729	0.0	0	4.9671	2	392.83	34.7
3	0.03237	0.0	0	6.0622	3	394.63	33.4
4	0.06905	0.0	0	6.0622	3	396.90	36.2

In [55]:
```
df_skewed.columns
```

Out[55]:
```
Index(['crim', 'zn', 'chas', 'dis', 'rad', 'black', 'medv'], dtype='object')
```

In [56]:
```
df_skewed.columns[(df_skewed[col]==0).any()]
```

Out[56]:
```
Index(['zn', 'chas'], dtype='object')
```

In [57]:
```
zn=[]

for i in df_skewed['zn']:
    if i==0:
        zn.append(1.0)
    else:
        zn.append(i)
```

In [58]:
```
chas=[]

for i in df_skewed['chas']:
    if i==0:
        chas.append(1.0)
    else:
        chas.append(i)
```

In [59]:
```
df_skewed['zn'] = zn
df_skewed['chas'] = chas
```

```
/tmp/ipykernel_6610/2309885715.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_skewed['zn'] = zn
/tmp/ipykernel_6610/2309885715.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_skewed['chas'] = chas
```

In [60]:
```
df_log = np.log(df_skewed)
```

In [61]:
```
df_log.head()
```

Out[61]:

	crim	zn	chas	dis	rad	black	medv
0	-5.064036	2.890372	0.0	1.408545	0.000000	5.983684	3.178054
1	-3.600502	0.000000	0.0	1.602836	0.693147	5.983684	3.072693
2	-3.601235	0.000000	0.0	1.602836	0.693147	5.973377	3.546740
3	-3.430523	0.000000	0.0	1.802073	1.098612	5.977949	3.508556
4	-2.672924	0.000000	0.0	1.802073	1.098612	5.983684	3.589059

In [62]:
```
df_log.columns
```

Out[62]:
```
Index(['crim', 'zn', 'chas', 'dis', 'rad', 'black', 'medv'], dtype='object')
```

In [63]:
```
unskewed_cols = df_skew[(df_skew.values>=-1) & (df_skew.values<=1)].index
```

In [64]:
```
unskewed_cols
```

Out[64]:
```
Index(['indus', 'nox', 'rm', 'age', 'tax', 'ptratio', 'lstat'], dtype='object')
```

In [65]:
```
df_unskewed = df_boston[unskewed_cols]
```

In [66]:
```
df_unskewed.columns
```

Out[66]:
```
Index(['indus', 'nox', 'rm', 'age', 'tax', 'ptratio', 'lstat'], dtype='object')
```

In [67]:
```
df_trans = pd.concat([df_log,df_unskewed],axis=1)
```

In [68]:
```
df_trans.head()
```

Out[68]:

	crim	zn	chas	dis	rad	black	medv	indus	nox	rm	age	tax	ptratio	lstat
0	-5.064036	2.890372	0.0	1.408545	0.000000	5.983684	3.178054	2.31	0.538	6.575	65.2	296	15.3	4.98
1	-3.600502	0.000000	0.0	1.602836	0.693147	5.983684	3.072693	7.07	0.469	6.421	78.9	242	17.8	9.14
2	-3.601235	0.000000	0.0	1.602836	0.693147	5.973377	3.546740	7.07	0.469	7.185	61.1	242	17.8	4.03
3	-3.430523	0.000000	0.0	1.802073	1.098612	5.977949	3.508556	2.18	0.458	6.998	45.8	222	18.7	2.94
4	-2.672924	0.000000	0.0	1.802073	1.098612	5.983684	3.589059	2.18	0.458	7.147	54.2	222	18.7	5.33

In [69]:
```
df_trans.columns
```

Out[69]:
```
Index(['crim', 'zn', 'chas', 'dis', 'rad', 'black', 'medv', 'indus', 'nox',
       'rm', 'age', 'tax', 'ptratio', 'lstat'],
      dtype='object')
```

In [70]:
```
y = df_trans['medv']
y.head()
```

Out[70]:
```
0    3.178054
1    3.072693
2    3.546740
3    3.508556
4    3.589059
Name: medv, dtype: float64
```

In [71]:
```
x = df_trans.drop('medv',axis=1)
x.head()
```

Out[71]:

	crim	zn	chas	dis	rad	black	indus	nox	rm	age	tax	ptratio	lstat
0	-5.064036	2.890372	0.0	1.408545	0.000000	5.983684	2.31	0.538	6.575	65.2	296	15.3	4.98
1	-3.600502	0.000000	0.0	1.602836	0.693147	5.983684	7.07	0.469	6.421	78.9	242	17.8	9.14
2	-3.601235	0.000000	0.0	1.602836	0.693147	5.973377	7.07	0.469	7.185	61.1	242	17.8	4.03
3	-3.430523	0.000000	0.0	1.802073	1.098612	5.977949	2.18	0.458	6.998	45.8	222	18.7	2.94
4	-2.672924	0.000000	0.0	1.802073	1.098612	5.983684	2.18	0.458	7.147	54.2	222	18.7	5.33

In [72]:
```
x_tr,x_test,y_tr,y_test = train_test_split(x,y,test_size=0.2,random_state=100)
```

In [73]:
```
model_lm_log = LinearRegression().fit(x_tr,y_tr)
```

In [74]:
```
p_log = model_lm_log.predict(x_test)
```

In [75]:
```
r2_score(y_test,p_log)
```

Out[75]:
```
0.7763324878210931
```

In [76]:
```
p_log
```

Out[76]:
```
array([3.43377188, 3.51470602, 3.03976385, 2.86225723, 3.00704382,
       3.22968289, 3.22835859, 3.13382666, 3.06944003, 2.85767449,
       3.30924033, 2.74392349, 2.96631084, 2.702732  , 3.71999238,
       3.11607728, 3.37242603, 2.98371583, 3.34767054, 3.7103211 ,
       3.50245786, 2.94003851, 2.98900088, 2.79955356, 2.56997228,
       2.43110776, 3.29634105, 2.80671475, 2.83468943, 2.89517395,
       2.78946315, 3.16521034, 3.71746226, 3.22611206, 3.31412356,
       3.40903011, 2.78215389, 2.70773384, 2.66476403, 3.17367748,
       3.0943514 , 3.05439562, 2.55257298, 3.07065281, 3.4524404 ,
       3.34063377, 2.81324924, 2.73063244, 2.73340838, 2.40623948,
       2.98668803, 2.97345976, 3.18405928, 3.24386919, 2.48941846,
       2.68423051, 3.34714069, 3.50781378, 2.31355703, 3.04956369,
       2.77638973, 2.86230375, 2.88488301, 3.30892801, 3.12176574,
       3.22470697, 2.79512041, 3.36155377, 3.0693455 , 3.08159544,
       2.77522579, 2.91604407, 2.22413151, 2.71542464, 3.38568358,
       2.242402  , 3.24357654, 3.53683308, 2.42338315, 3.24181196,
       3.55541203, 3.71314339, 2.6357082 , 2.72673612, 2.78704307,
       2.58739689, 3.00592054, 3.14295472, 2.497522  , 2.84902196,
       3.45344434, 3.13516702, 3.16621207, 3.21041223, 2.86916129,
       2.95157862, 3.06341537, 3.65612332, 2.75375733, 3.21958463,
       2.64430837, 2.59992191])
```

In [77]:
```
p_actual = np.exp(p_log)
y_actual = np.exp(y_test)
```

In [78]:
```
r2_score(y_actual,p_actual)
```

Out[78]:
```
0.7874673188199536
```

Model 6

Ridge Regression

In [79]:
```
x = df_boston.iloc[:,0:13]
y = df_boston.iloc[:,13]
```

In [80]:
```
from mlxtend.evaluate import bias_variance_decomp
```

In [81]:
```
x_tr,x_test,y_tr,y_test = train_test_split(x,y,test_size=0.2,random_state=100)
```

In [82]:
```
model_lm=LinearRegression()
```

In [83]:
```
mse,bias,variance = bias_variance_decomp(model_lm,x_tr.values,y_tr.values,
                                         x_test.values,y_test.values,
                                         loss='mse',random_seed=1,num_rounds=200)
```

In [113…
```
print("mse",mse)
print("bias",bias)
print("variance",variance)
```

```
mse 24.994701267641286
bias 23.72261531485006
variance 1.2720859527912267
```

In [85]:
```
model_lm.fit(x_tr,y_tr)
```

Out[85]:
```
LinearRegression()
```

In [86]:
```
p_lm=model_lm.predict(x_test)
```

In [87]:
```
r2_score(y_test,p_lm)
```

Out[87]:
```
0.7555033086871303
```

In [88]:
```
model_lm.coef_
```

Out[88]:
```
array([-8.14896492e-02,  4.80407782e-02, -5.47150249e-03,  3.06260576e+00,
       -1.61368815e+01,  3.67245067e+00, -8.51525259e-03, -1.51740854e+00,
        2.87271007e-01, -1.21207598e-02, -9.24160757e-01,  9.53460812e-03,
       -4.85895548e-01])
```

In [89]:
```
x_tr.columns
```

Out[89]:
```
Index(['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax',
       'ptratio', 'black', 'lstat'],
      dtype='object')
```

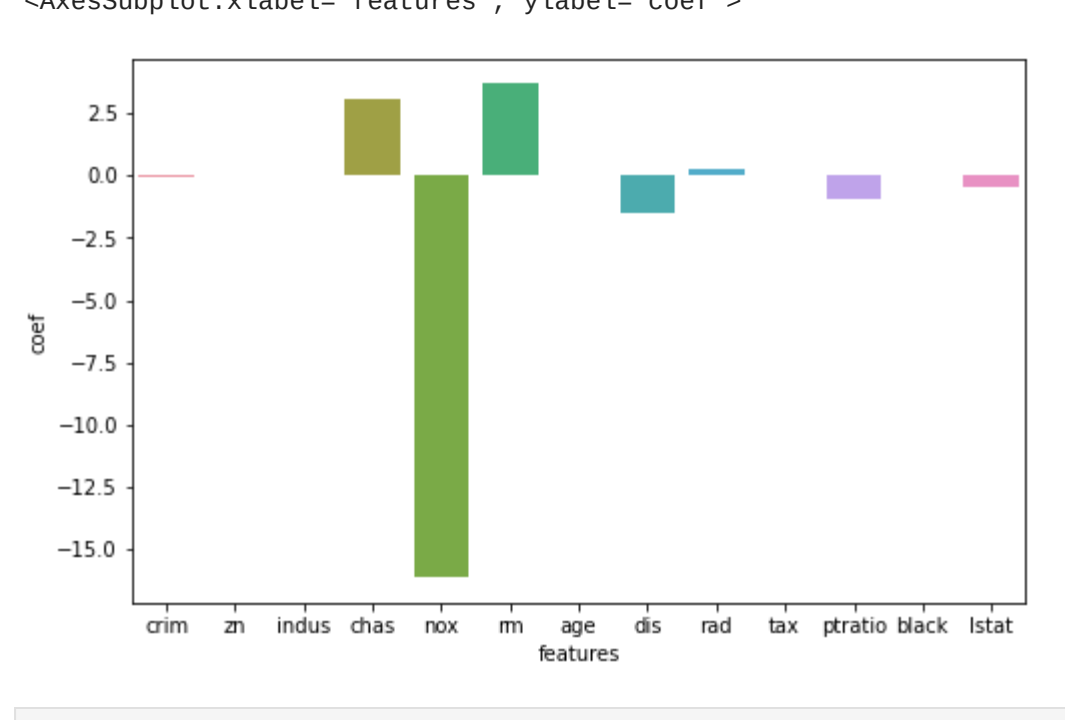
In [90]:
```
lm_coef=pd.DataFrame({"features":x_tr.columns,"coef":model_lm.coef_})
```

In [91]:
```
lm_coef
```

Out[91]:

	features	coef
0	crim	-0.081490
1	zn	0.048041
2	indus	-0.005472
3	chas	3.062606
4	nox	-16.136881
5	rm	3.672451
6	age	-0.008515
7	dis	-1.517409
8	rad	0.287271
9	tax	-0.012121
10	ptratio	-0.924161
11	black	0.009535
12	lstat	-0.485896

In [92]:
```
plt.figure(figsize=(8,5))
sns.barplot(x="features",y="coef",data=lm_coef)
```
Out[92]: <AxesSubplot:xlabel='features', ylabel='coef'>



In [93]:
```
from sklearn.linear_model import Ridge
model_ridge=Ridge(alpha=.001)
```

In [94]:
```
mse,bias,variance = bias_variance_decomp(model_ridge,x_tr.values,y_tr.values,
                                        x_test.values,y_test.values,
                                        loss='mse',random_seed=1,num_rounds=200)
```

In [95]:
```
print("mse",mse)
print("bias",bias)
print("variance",variance)
```
```
mse 24.99353949601843
bias 23.72159314480973
variance 1.2719463512087081
```

In [96]:
```
model_ridge.fit(x_tr,y_tr)
```
Out[96]: Ridge(alpha=0.001)

In [97]:
```
p_ridge=model_ridge.predict(x_test)
```

In [98]:
```
r2_score(y_test,p_ridge)
```
Out[98]: 0.7554964592521939

In [99]:
```
model_ridge.coef_
```
Out[99]:
```
array([-8.14815824e-02,  4.80427788e-02, -5.52275128e-03,  3.06238531e+00,
       -1.61238047e+01,  3.67254369e+00, -8.52714530e-03, -1.51721487e+00,
        2.87238523e-01, -1.21220233e-02, -9.24018399e-01,  9.53529307e-03,
       -4.85908241e-01])
```

In [100…
```
ridge_coef=pd.DataFrame({"features":x_tr.columns,"coef":model_ridge.coef_})
```

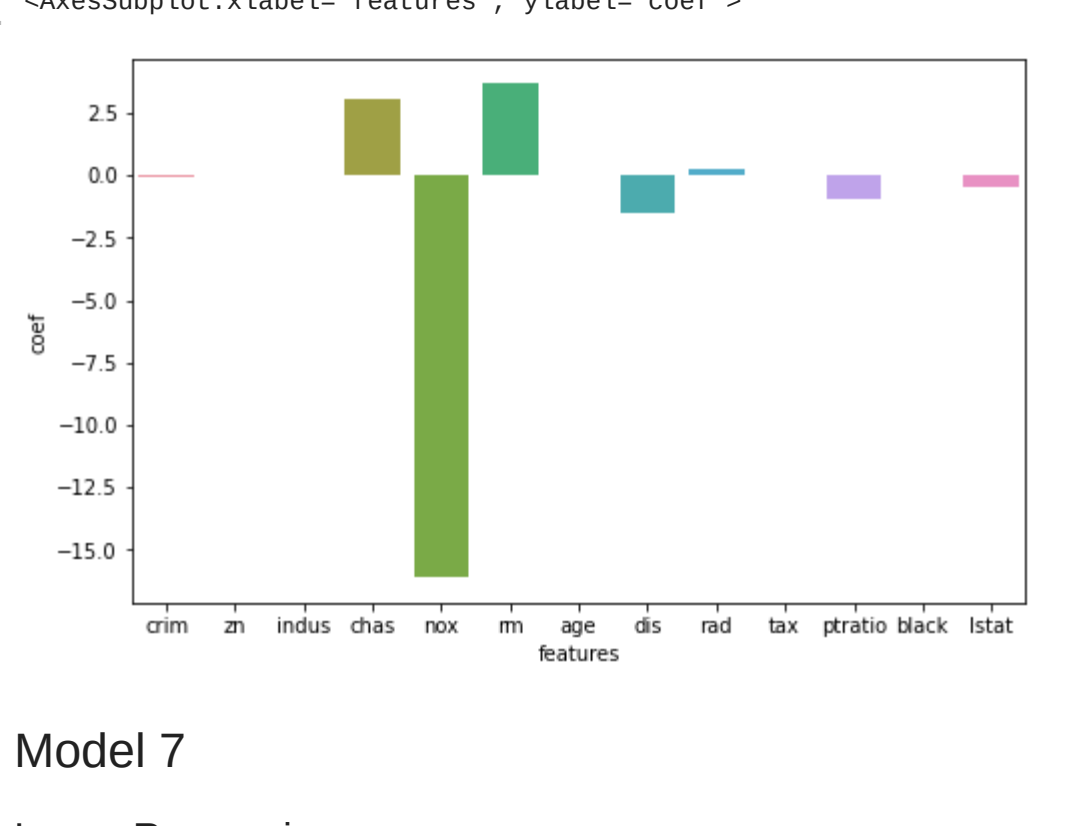
In [101…
```
ridge_coef
```
Out[101…

	features	coef
0	crim	-0.081482
1	zn	0.048043
2	indus	-0.005523
3	chas	3.062385
4	nox	-16.123805
5	rm	3.672544
6	age	-0.008527
7	dis	-1.517215
8	rad	0.287239
9	tax	-0.012122
10	ptratio	-0.924018
11	black	0.009535
12	lstat	-0.485908

Plot the coef of each features

In [ ]:

In [102…
```
plt.figure(figsize=(8,5))
sns.barplot(x="features",y="coef",data=ridge_coef)
```
Out[102…: <AxesSubplot:xlabel='features', ylabel='coef'>



Model 7

Lasso Regression

In [103…
```
from sklearn.linear_model import Lasso
```

In [104…
```
model_lasso=Lasso(alpha=.0001)
```

In [105…
```
mse,bias,variance = bias_variance_decomp(model_lasso,x_tr.values,y_tr.values,
                                        x_test.values,y_test.values,
                                        loss='mse',random_seed=1,num_rounds=200)
```

In [106…
```
print("mse",mse)
print("bias",bias)
print("variance",variance)
```
```
mse 24.994701267641286
bias 23.72261531485006
variance 1.2720859527912267
```

In [107…
```
model_lasso.fit(x_tr,y_tr)
```
Out[107…: Lasso(alpha=0.0001)

In [108…
```
p_lasso = model_lasso.predict(x_test)
```

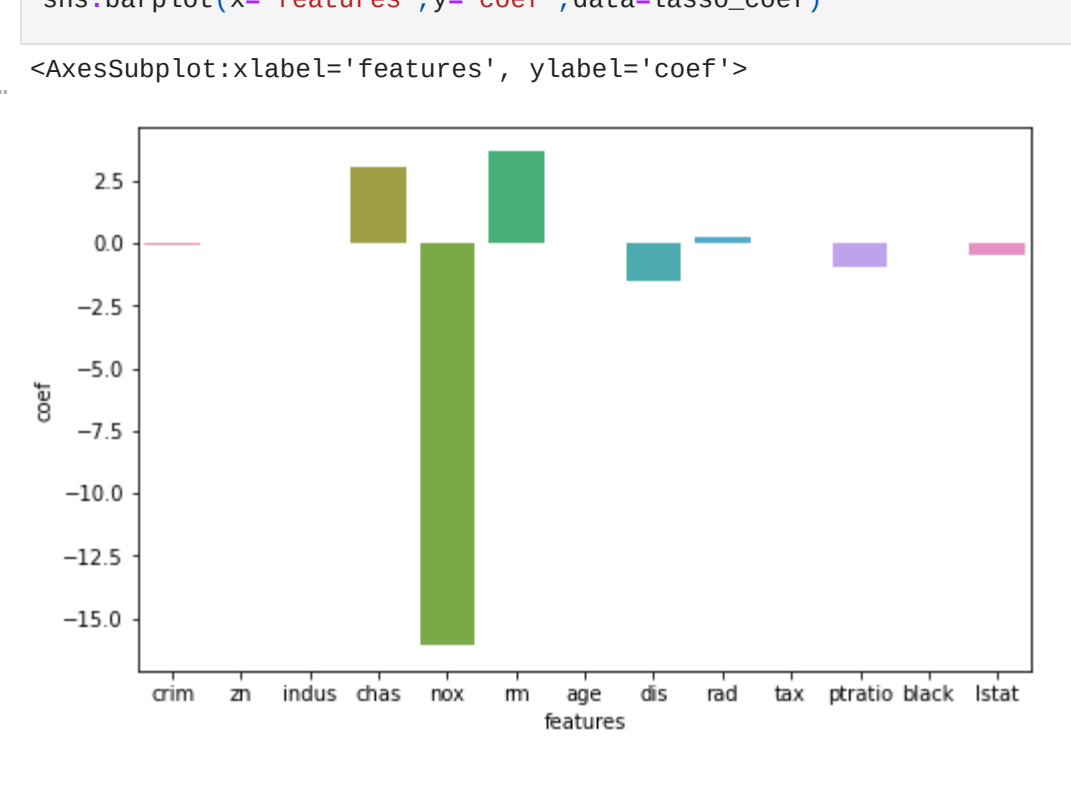
In [109…
```
r2_score(y_test,p_lasso)
```
Out[109…: 0.7554874991489418

In [110…
```
lasso_coef = pd.DataFrame({"features":x_tr.columns,"coef":model_lasso.coef_})
```

In [111…
```
lasso_coef
```
Out[111…

	features	coef
0	crim	-0.081467
1	zn	0.048046
2	indus	-0.005585
3	chas	3.060885
4	nox	-16.103443
5	rm	3.672472
6	age	-0.008541
7	dis	-1.516857
8	rad	0.287198
9	tax	-0.012125
10	ptratio	-0.923802
11	black	0.009536
12	lstat	-0.485950

In [112…
```
plt.figure(figsize=(8,5))
sns.barplot(x="features",y="coef",data=lasso_coef)
```
Out[112…: <AxesSubplot:xlabel='features', ylabel='coef'>



Regression-tree,Assignment-9

Sumon Singh,M.Sc Computer Science Specialization in Data Science

22/12/2021

Regression tree on CarPrice dataset

This is a r-script which performs regression tree on CarPrice dataset. It generates a tree based model. The model evaluates the price of cars based on the features. The goodness of the model has been evaluated using the r2-score and root-mean-square-error.

Load Libraries

```
library(rpart)
library(rpart.plot)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(Metrics)
```

Import Dataset and transform dataset

```
df = read.csv('/home/sumon/Desktop/Datasets/CarPrice_Assignment (1).csv')
head(df)
```

```
##   car_ID symboling                  CarName fueltype aspiration doornumber
## 1      1         3       alfa-romero giulia      gas        std        two
## 2      2         3      alfa-romero stelvio      gas        std        two
## 3      3         1 alfa-romero Quadrifoglio      gas        std        two
## 4      4         2              audi 100 ls      gas        std       four
```

```
## 5      5         2              audi 100ls      gas        std       four
## 6      6         2                audi fox      gas        std        two
##       carbody drivewheel enginelocation wheelbase carlength carwidth carheight
## 1 convertible        rwd          front      88.6     168.8     64.1      48.8
## 2 convertible        rwd          front      88.6     168.8     64.1      48.8
## 3   hatchback        rwd          front      94.5     171.2     65.5      52.4
## 4       sedan        fwd          front      99.8     176.6     66.2      54.3
## 5       sedan        4wd          front      99.4     176.6     66.4      54.3
## 6       sedan        fwd          front      99.8     177.3     66.3      53.1
##   curbweight enginetype cylindernumber enginesize fuelsystem boreratio stroke
## 1       2548       dohc           four        130       mpfi      3.47   2.68
## 2       2548       dohc           four        130       mpfi      3.47   2.68
## 3       2823       ohcv            six        152       mpfi      2.68   3.47
## 4       2337        ohc           four        109       mpfi      3.19   3.40
## 5       2824        ohc           five        136       mpfi      3.19   3.40
## 6       2507        ohc           five        136       mpfi      3.19   3.40
##   compressionratio horsepower peakrpm citympg highwaympg price
## 1              9.0        111    5000      21         27 13495
## 2              9.0        111    5000      21         27 16500
## 3              9.0        154    5000      19         26 16500
## 4             10.0        102    5500      24         30 13950
## 5              8.0        115    5500      18         22 17450
## 6              8.5        110    5500      19         25 15250
```

```
glimpse(df)
```

```
## Rows: 205
## Columns: 26
## $ car_ID           <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16~
## $ symboling        <int> 3, 3, 1, 2, 2, 2, 1, 1, 1, 0, 2, 0, 0, 0, 1, 0, 0, 0,~
## $ CarName          <fct> alfa-romero giulia, alfa-romero stelvio, alfa-romero ~
## $ fueltype         <fct> gas, gas, gas, gas, gas, gas, gas, gas, gas, gas, gas~
## $ aspiration       <fct> std, std, std, std, std, std, std, std, turbo, turbo,~
## $ doornumber       <fct> two, two, two, four, four, two, four, four, four, two~
## $ carbody          <fct> convertible, convertible, hatchback, sedan, sedan, se~
## $ drivewheel       <fct> rwd, rwd, rwd, fwd, 4wd, fwd, fwd, fwd, fwd, 4wd, rwd~
## $ enginelocation   <fct> front, front, front, front, front, front, front, fron~
## $ wheelbase        <dbl> 88.6, 88.6, 94.5, 99.8, 99.4, 99.8, 105.8, 105.8, 105~
## $ carlength        <dbl> 168.8, 168.8, 171.2, 176.6, 176.6, 177.3, 192.7, 192.~
## $ carwidth         <dbl> 64.1, 64.1, 65.5, 66.2, 66.4, 66.3, 71.4, 71.4, 71.4,~
## $ carheight        <dbl> 48.8, 48.8, 52.4, 54.3, 54.3, 53.1, 55.7, 55.7, 55.9,~
## $ curbweight       <int> 2548, 2548, 2823, 2337, 2824, 2507, 2844, 2954, 3086,~
## $ enginetype       <fct> dohc, dohc, ohcv, ohc, ohc, ohc, ohc, ohc, ohc, ohc, ~
## $ cylindernumber   <fct> four, four, six, four, five, five, five, five, five, ~
## $ enginesize       <int> 130, 130, 152, 109, 136, 136, 136, 136, 131, 131, 108~
## $ fuelsystem       <fct> mpfi, mpfi, mpfi, mpfi, mpfi, mpfi, mpfi, mpfi, mpfi,~
## $ boreratio        <dbl> 3.47, 3.47, 2.68, 3.19, 3.19, 3.19, 3.19, 3.19, 3.13,~
## $ stroke           <dbl> 2.68, 2.68, 3.47, 3.40, 3.40, 3.40, 3.40, 3.40, 3.40,~
## $ compressionratio <dbl> 9.00, 9.00, 9.00, 10.00, 8.00, 8.50, 8.50, 8.50, 8.30~
## $ horsepower       <int> 111, 111, 154, 102, 115, 110, 110, 110, 140, 160, 101~
## $ peakrpm          <int> 5000, 5000, 5000, 5500, 5500, 5500, 5500, 5500, 5500,~
## $ citympg          <int> 21, 21, 19, 24, 18, 19, 19, 19, 17, 16, 23, 23, 21, 2~
## $ highwaympg       <int> 27, 27, 26, 30, 22, 25, 25, 25, 20, 22, 29, 29, 28, 2~
## $ price            <dbl> 13495.00, 16500.00, 16500.00, 13950.00, 17450.00, 152~
```

```
table(df$symboling)
```

```
##
## -2 -1  0  1  2  3
##  3 22 67 54 32 27
```

```
df$symboling=as.factor(df$symboling)
df$cylindernumber = as.factor(df$cylindernumber)
colnames(df)
```

```
##  [1] "car_ID"           "symboling"        "CarName"          "fueltype"
##  [5] "aspiration"       "doornumber"       "carbody"          "drivewheel"
##  [9] "enginelocation"   "wheelbase"        "carlength"        "carwidth"
## [13] "carheight"        "curbweight"       "enginetype"       "cylindernumber"
## [17] "enginesize"       "fuelsystem"       "boreratio"        "stroke"
## [21] "compressionratio" "horsepower"       "peakrpm"          "citympg"
## [25] "highwaympg"       "price"
```

```
nrow(df)
```

```
## [1] 205
```

```
df = df[,-1]
head(df)
```

```
##   symboling                  CarName fueltype aspiration doornumber     carbody
## 1         3       alfa-romero giulia      gas        std        two convertible
## 2         3      alfa-romero stelvio      gas        std        two convertible
## 3         1 alfa-romero Quadrifoglio      gas        std        two   hatchback
## 4         2              audi 100 ls      gas        std       four       sedan
## 5         2               audi 100ls      gas        std       four       sedan
## 6         2                 audi fox      gas        std        two       sedan
##   drivewheel enginelocation wheelbase carlength carwidth carheight curbweight
## 1        rwd          front      88.6     168.8     64.1      48.8       2548
## 2        rwd          front      88.6     168.8     64.1      48.8       2548
## 3        rwd          front      94.5     171.2     65.5      52.4       2823
## 4        fwd          front      99.8     176.6     66.2      54.3       2337
## 5        4wd          front      99.4     176.6     66.4      54.3       2824
## 6        fwd          front      99.8     177.3     66.3      53.1       2507
##   enginetype cylindernumber enginesize fuelsystem boreratio stroke
## 1       dohc           four        130       mpfi      3.47   2.68
## 2       dohc           four        130       mpfi      3.47   2.68
## 3       ohcv            six        152       mpfi      2.68   3.47
## 4        ohc           four        109       mpfi      3.19   3.40
## 5        ohc           five        136       mpfi      3.19   3.40
## 6        ohc           five        136       mpfi      3.19   3.40
##   compressionratio horsepower peakrpm citympg highwaympg price
## 1              9.0        111    5000      21         27 13495
## 2              9.0        111    5000      21         27 16500
## 3              9.0        154    5000      19         26 16500
## 4             10.0        102    5500      24         30 13950
## 5              8.0        115    5500      18         22 17450
## 6              8.5        110    5500      19         25 15250
```

Sampling

```
s=sample(nrow(df),0.8*(nrow(df)))
df_tr = df[s,]
df_test = df[-s,]
nrow(df_test)
```

```
## [1] 41
```

Regression Tree

```
tree = rpart(price~.,data = df_tr,method = "anova",control = rpart.control(minsplit = 2,
                                                                            minbucket = 1,
                                                                            maxdepth = 20))
tree
```

```
## n= 164
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 164 10130040000 13364.110
##    2) CarName=alfa-romero giulia,alfa-romero Quadrifoglio,alfa-romero stelvio,audi 100 ls,audi 100ls
##      4) CarName=chevrolet impala,chevrolet monte carlo,chevrolet vega 2300,dodge challenger se,dodge
##        8) curbweight< 2291.5 54    63021420  7059.306 *
##        9) curbweight>=2291.5 42   135257900 10322.790 *
##      5) CarName=alfa-romero giulia,alfa-romero Quadrifoglio,alfa-romero stelvio,audi 100 ls,audi 100l
##       10) CarName=alfa-romero giulia,audi 100 ls,audi fox,dodge coronet custom (sw),mazda glc,mazda
##       11) CarName=alfa-romero Quadrifoglio,alfa-romero stelvio,audi 100ls,audi 5000,audi 5000s (dies
##    3) CarName=audi 4000,bmw x1,bmw x3,bmw x5,bmw z4,buick century,buick century luxus (sw),buick ele
##      6) enginesize< 188.5 12   152626400 24135.330
##       12) CarName=audi 4000,bmw x1,bmw x3,bmw z4,porsche macan,volvo 246,volvo 264gl 8    25384740 2
##       13) CarName=buick century,buick century luxus (sw),buick electra 225 custom,buick skyhawk 4
##      7) enginesize>=188.5 10   173070500 36056.350
##       14) CarName=bmw x3,buick opel isuzu deluxe,jaguar xf,jaguar xj,porcshce panamera,porsche boxte
##       15) CarName=bmw x5,buick regal sport coupe (turbo) 2     8343612 43357.500 *
```

```
summary(tree)
```

```
## Call:
## rpart(formula = price ~ ., data = df_tr, method = "anova", control = rpart.control(minsplit = 2,
##     minbucket = 1, maxdepth = 20))
##   n= 164
##
##           CP nsplit  rel error    xerror       xstd
## 1 0.65743617      0 1.00000000 1.0165407 0.17575178
## 2 0.16414463      1 0.34256383 0.4230513 0.05068405
## 3 0.07651984      2 0.17841920 0.2740589 0.04337566
## 4 0.02483834      3 0.10189937 0.2753442 0.04424943
```

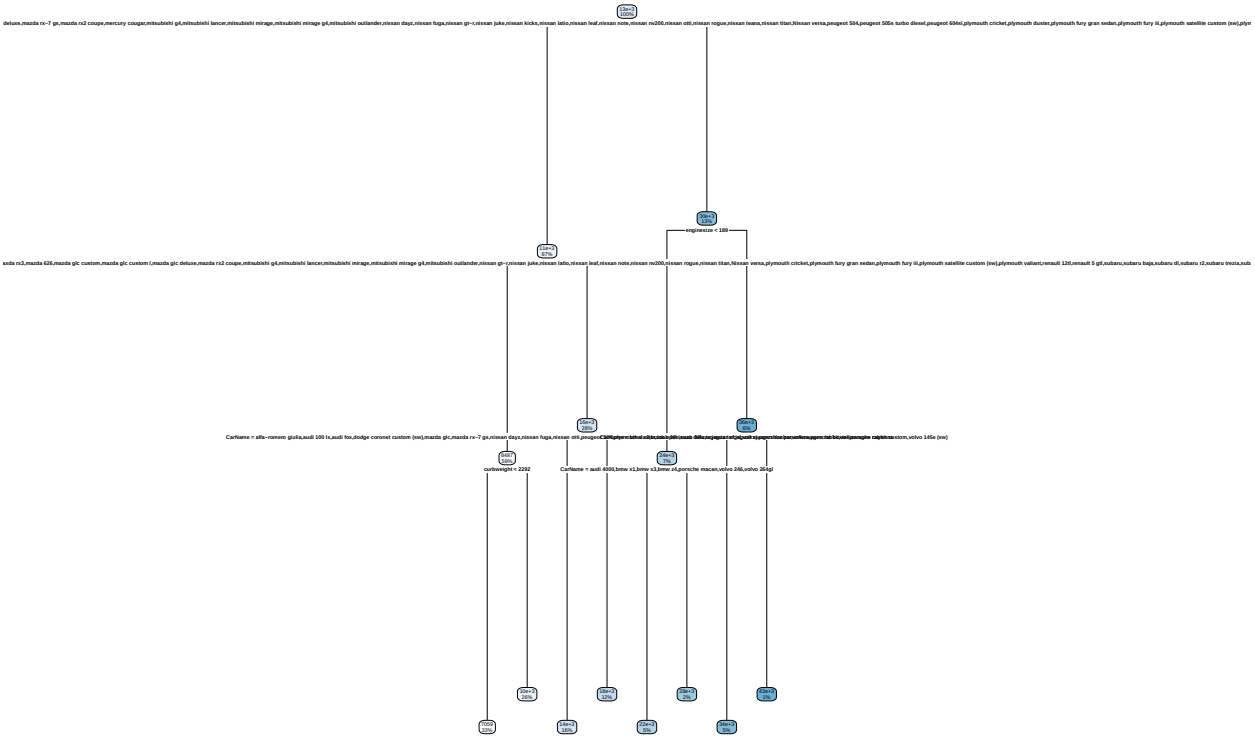
```
## 5 0.01315562      4 0.07706102 0.2043802 0.03460840
## 6 0.01074206      5 0.06390540 0.1976978 0.03245038
## 7 0.01015521      6 0.05316334 0.1935490 0.03191049
## 8 0.01000000      7 0.04300813 0.1935490 0.03191049
##
## Variable importance
##    CarName enginesize curbweight horsepower  carlength   carwidth    citympg
##         26         16         13          9          9          9          5
##  wheelbase highwaympg enginetype aspiration  carheight
##          4          4          2          1          1
##
## Node number 1: 164 observations,    complexity param=0.6574362
##   mean=13364.11, MSE=6.176853e+07
##   left son=2 (142 obs) right son=3 (22 obs)
##   Primary splits:
##       CarName        splits as  LLLLLRLLL-RR-RRRR-RRRR-LLLLLLLL-LLLLLLLL--LLL-LRR--LLL-LLL-LLLLLLL-L
##       enginesize     < 182    to the left,  improve=0.6351908, (0 missing)
##       cylindernumber splits as  RRLRL-L, improve=0.5299763, (0 missing)
##       curbweight     < 2745   to the left,  improve=0.5291598, (0 missing)
##       horsepower     < 118    to the left,  improve=0.5144319, (0 missing)
##   Surrogate splits:
##       enginesize < 182    to the left,  agree=0.951, adj=0.636, (0 split)
##       carwidth   < 68.85  to the left,  agree=0.921, adj=0.409, (0 split)
##       curbweight < 3331   to the left,  agree=0.921, adj=0.409, (0 split)
##       horsepower < 175.5  to the left,  agree=0.915, adj=0.364, (0 split)
##       carlength  < 188.9  to the left,  agree=0.902, adj=0.273, (0 split)
##
## Node number 2: 142 observations,    complexity param=0.1641446
##   mean=10855.82, MSE=1.668549e+07
##   left son=4 (96 obs) right son=5 (46 obs)
##   Primary splits:
##       CarName    splits as  RRRRR-RRR--------------LLLLLLLR-LLLLLLLL--LLL-L----LLR-LLL-RLRLLLL-L--RRR
##       curbweight < 2542   to the left,  improve=0.6619698, (0 missing)
##       highwaympg < 28.5   to the right, improve=0.6350272, (0 missing)
##       horsepower < 94.5   to the left,  improve=0.5819708, (0 missing)
##       carwidth   < 65.55  to the left,  improve=0.5645413, (0 missing)
##   Surrogate splits:
##       highwaympg < 28.5   to the right, agree=0.908, adj=0.717, (0 split)
##       curbweight < 2542   to the left,  agree=0.894, adj=0.674, (0 split)
##       carlength  < 178    to the left,  agree=0.887, adj=0.652, (0 split)
##       wheelbase  < 98.95  to the left,  agree=0.880, adj=0.630, (0 split)
##       citympg    < 22     to the right, agree=0.873, adj=0.609, (0 split)
##
## Node number 3: 22 observations,    complexity param=0.07651984
##   mean=29553.98, MSE=5.003845e+07
##   left son=6 (12 obs) right son=7 (10 obs)
##   Primary splits:
##       enginesize < 188.5  to the left,  improve=0.7041394, (0 missing)
##       horsepower < 169    to the left,  improve=0.6230080, (0 missing)
##       CarName    splits as  -----L----LL-RLLL-LRRR-------------------------RR----------------------
##       citympg    < 16.5   to the right, improve=0.4847435, (0 missing)
##       carlength  < 193.25 to the left,  improve=0.4815738, (0 missing)
##   Surrogate splits:
##       horsepower < 149    to the left,  agree=0.955, adj=0.9, (0 split)
```

```
##       enginetype splits as  RR-LRR-,    agree=0.909, adj=0.8, (0 split)
##       citympg    < 18     to the right, agree=0.909, adj=0.8, (0 split)
##       aspiration splits as  RL,         agree=0.818, adj=0.6, (0 split)
##       carheight  < 54     to the right, agree=0.818, adj=0.6, (0 split)
##
## Node number 4: 96 observations,    complexity param=0.02483834
##   mean=8487.078, MSE=4686383
##   left son=8 (54 obs) right son=9 (42 obs)
##   Primary splits:
##       curbweight < 2291.5 to the left,  improve=0.5592742, (0 missing)
##       horsepower < 83     to the left,  improve=0.5127302, (0 missing)
##       carlength  < 168    to the left,  improve=0.4878257, (0 missing)
##       carwidth   < 64.5   to the left,  improve=0.4860642, (0 missing)
##       wheelbase  < 95.8   to the left,  improve=0.4597126, (0 missing)
##   Surrogate splits:
##       carlength  < 168.85 to the left,  agree=0.917, adj=0.810, (0 split)
##       carwidth   < 64.5   to the left,  agree=0.906, adj=0.786, (0 split)
##       wheelbase  < 95.8   to the left,  agree=0.896, adj=0.762, (0 split)
##       CarName    splits as  ----------------------LLLLLLL--RLLRLLLR--LRL-R----LR--RRL--L-RLLR-R----
##       enginesize < 109.5  to the left,  agree=0.885, adj=0.738, (0 split)
##
## Node number 5: 46 observations,    complexity param=0.01015521
##   mean=15799.29, MSE=5579454
##   left son=10 (26 obs) right son=11 (20 obs)
##   Primary splits:
##       CarName    splits as  LRRLR-RRL---------------------L----------------------L-----L-R--------LL
##       curbweight < 2820.5 to the left,  improve=0.2741386, (0 missing)
##       carwidth   < 67.45  to the left,  improve=0.2430708, (0 missing)
##       horsepower < 158    to the left,  improve=0.1918464, (0 missing)
##       symboling  splits as  LRLRRL, improve=0.1772632, (0 missing)
##   Surrogate splits:
##       carwidth   < 67.05  to the left,  agree=0.739, adj=0.40, (0 split)
##       symboling  splits as  LRLRLL,     agree=0.717, adj=0.35, (0 split)
##       curbweight < 2820.5 to the left,  agree=0.696, adj=0.30, (0 split)
##       enginesize < 125.5  to the left,  agree=0.696, adj=0.30, (0 split)
##       horsepower < 153    to the left,  agree=0.696, adj=0.30, (0 split)
##
## Node number 6: 12 observations,    complexity param=0.01074206
##   mean=24135.33, MSE=1.271887e+07
##   left son=12 (8 obs) right son=13 (4 obs)
##   Primary splits:
##       CarName    splits as  -----L----LL--LRR-R--R-------------------------------------------------
##       curbweight < 3356   to the left,  improve=0.7129660, (0 missing)
##       enginesize < 173.5  to the left,  improve=0.7129660, (0 missing)
##       stroke     < 3.52   to the left,  improve=0.7129660, (0 missing)
##       carwidth   < 69.6   to the left,  improve=0.6320828, (0 missing)
##   Surrogate splits:
##       curbweight < 3356   to the left,  agree=1.000, adj=1.00, (0 split)
##       enginesize < 173.5  to the left,  agree=1.000, adj=1.00, (0 split)
##       stroke     < 3.52   to the left,  agree=1.000, adj=1.00, (0 split)
##       wheelbase  < 109.55 to the left,  agree=0.917, adj=0.75, (0 split)
##       carwidth   < 69.6   to the left,  agree=0.917, adj=0.75, (0 split)
##
## Node number 7: 10 observations,    complexity param=0.01315562
```

```
##   mean=36056.35, MSE=1.730705e+07
##   left son=14 (8 obs) right son=15 (2 obs)
##   Primary splits:
##       CarName          splits as  ----------L-R-----LR--------------------------LL-----------------
##       compressionratio < 8.05   to the right, improve=0.6547290, (0 missing)
##       peakrpm          < 4625   to the right, improve=0.5604897, (0 missing)
##       citympg          < 14.5   to the right, improve=0.5604897, (0 missing)
##       highwaympg       < 17     to the right, improve=0.5604897, (0 missing)
##   Surrogate splits:
##       compressionratio < 8.05   to the right, agree=0.9, adj=0.5, (0 split)
##
## Node number 8: 54 observations
##   mean=7059.306, MSE=1167063
##
## Node number 9: 42 observations
##   mean=10322.79, MSE=3220427
##
## Node number 10: 26 observations
##   mean=14487.69, MSE=3902651
##
## Node number 11: 20 observations
##   mean=17504.36, MSE=2615665
##
## Node number 12: 8 observations
##   mean=22006, MSE=3173092
##
## Node number 13: 4 observations
##   mean=28394, MSE=4606060
##
## Node number 14: 8 observations
##   mean=34231.06, MSE=3932486
##
## Node number 15: 2 observations
##   mean=43357.5, MSE=4171806
```

Including Plots

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Prediction

```
p = predict(tree,df_test)
data_frame("Actual"=df_test$price,"Predicted"=p)
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

```
## # A tibble: 41 x 2
##    Actual Predicted
##     <dbl>     <dbl>
##  1 16430     10323.
##  2 16925     10323.
##  3 30760     43358.
##  4  7957      7059.
##  5  7295      7059.
##  6  7895      7059.
##  7  9095      7059.
##  8  8916.     7059.
##  9 36000     34231.
## 10  6095      7059.
## # ... with 31 more rows
```

Effeciency

```
rmse(df_test$price,p)
```

```
## [1] 3651.653
```

```
r2_score = (cor(df_test$price,p))**2
r2_score
```

```
## [1] 0.8449082
```

Note : r2-square value is approximately 74.5% that means the car price explaines 74.5% variation of features.

Logistic Regression,Assignment-10

Sumon Singh,M.Sc Computer Science Specialization in Data Science

01/01/2022

R Markdown

Perform logistic regression on breast_cancer dataset

Import dataset and explore

```
df_c = read.csv('/home/sumon/Desktop/Datasets/breast_cancer.csv')
head(df_c)
```

```
##         id diagnosis radius_mean texture_mean perimeter_mean area_mean
## 1   842302         M       17.99        10.38         122.80    1001.0
## 2   842517         M       20.57        17.77         132.90    1326.0
## 3 84300903         M       19.69        21.25         130.00    1203.0
## 4 84348301         M       11.42        20.38          77.58     386.1
## 5 84358402         M       20.29        14.34         135.10    1297.0
## 6   843786         M       12.45        15.70          82.57     477.1
##   smoothness_mean compactness_mean concavity_mean concave.points_mean
## 1         0.11840          0.27760         0.3001             0.14710
## 2         0.08474          0.07864         0.0869             0.07017
## 3         0.10960          0.15990         0.1974             0.12790
## 4         0.14250          0.28390         0.2414             0.10520
## 5         0.10030          0.13280         0.1980             0.10430
## 6         0.12780          0.17000         0.1578             0.08089
##   symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se
## 1        0.2419                0.07871    1.0950     0.9053        8.589
## 2        0.1812                0.05667    0.5435     0.7339        3.398
## 3        0.2069                0.05999    0.7456     0.7869        4.585
## 4        0.2597                0.09744    0.4956     1.1560        3.445
## 5        0.1809                0.05883    0.7572     0.7813        5.438
## 6        0.2087                0.07613    0.3345     0.8902        2.217
##   area_se smoothness_se compactness_se concavity_se concave.points_se
## 1  153.40      0.006399        0.04904      0.05373           0.01587
## 2   74.08      0.005225        0.01308      0.01860           0.01340
## 3   94.03      0.006150        0.04006      0.03832           0.02058
## 4   27.23      0.009110        0.07458      0.05661           0.01867
## 5   94.44      0.011490        0.02461      0.05688           0.01885
## 6   27.19      0.007510        0.03345      0.03672           0.01137
##   symmetry_se fractal_dimension_se radius_worst texture_worst perimeter_worst
## 1     0.03003             0.006193        25.38         17.33          184.60
```

```
## 2     0.01389             0.003532        24.99         23.41          158.80
## 3     0.02250             0.004571        23.57         25.53          152.50
## 4     0.05963             0.009208        14.91         26.50           98.87
## 5     0.01756             0.005115        22.54         16.67          152.20
## 6     0.02165             0.005082        15.47         23.75          103.40
##   area_worst smoothness_worst compactness_worst concavity_worst
## 1     2019.0           0.1622            0.6656          0.7119
## 2     1956.0           0.1238            0.1866          0.2416
## 3     1709.0           0.1444            0.4245          0.4504
## 4      567.7           0.2098            0.8663          0.6869
## 5     1575.0           0.1374            0.2050          0.4000
## 6      741.6           0.1791            0.5249          0.5355
##   concave.points_worst symmetry_worst fractal_dimension_worst
## 1               0.2654         0.4601                 0.11890
## 2               0.1860         0.2750                 0.08902
## 3               0.2430         0.3613                 0.08758
## 4               0.2575         0.6638                 0.17300
## 5               0.1625         0.2364                 0.07678
## 6               0.1741         0.3985                 0.12440
```

```
colnames(df_c)
```

```
##  [1] "id"                     "diagnosis"
##  [3] "radius_mean"            "texture_mean"
##  [5] "perimeter_mean"         "area_mean"
##  [7] "smoothness_mean"        "compactness_mean"
##  [9] "concavity_mean"         "concave.points_mean"
## [11] "symmetry_mean"          "fractal_dimension_mean"
## [13] "radius_se"              "texture_se"
## [15] "perimeter_se"           "area_se"
## [17] "smoothness_se"          "compactness_se"
## [19] "concavity_se"           "concave.points_se"
## [21] "symmetry_se"            "fractal_dimension_se"
## [23] "radius_worst"           "texture_worst"
## [25] "perimeter_worst"        "area_worst"
## [27] "smoothness_worst"       "compactness_worst"
## [29] "concavity_worst"        "concave.points_worst"
## [31] "symmetry_worst"         "fractal_dimension_worst"
```

```
dim(df_c)
```

```
## [1] 569  32
```

```
str(df_c)
```

```
## 'data.frame':    569 obs. of  32 variables:
##  $ id                     : int  842302 842517 84300903 84348301 84358402 843786 844359 84458202 844
##  $ diagnosis              : Factor w/ 2 levels "B","M": 2 2 2 2 2 2 2 2 2 2 ...
##  $ radius_mean            : num  18 20.6 19.7 11.4 20.3 ...
##  $ texture_mean           : num  10.4 17.8 21.2 20.4 14.3 ...
##  $ perimeter_mean         : num  122.8 132.9 130 77.6 135.1 ...
##  $ area_mean              : num  1001 1326 1203 386 1297 ...
```

```
##  $ smoothness_mean        : num  0.1184 0.0847 0.1096 0.1425 0.1003 ...
##  $ compactness_mean       : num  0.2776 0.0786 0.1599 0.2839 0.1328 ...
##  $ concavity_mean         : num  0.3001 0.0869 0.1974 0.2414 0.198 ...
##  $ concave.points_mean    : num  0.1471 0.0702 0.1279 0.1052 0.1043 ...
##  $ symmetry_mean          : num  0.242 0.181 0.207 0.26 0.181 ...
##  $ fractal_dimension_mean : num  0.0787 0.0567 0.06 0.0974 0.0588 ...
##  $ radius_se              : num  1.095 0.543 0.746 0.496 0.757 ...
##  $ texture_se             : num  0.905 0.734 0.787 1.156 0.781 ...
##  $ perimeter_se           : num  8.59 3.4 4.58 3.44 5.44 ...
##  $ area_se                : num  153.4 74.1 94 27.2 94.4 ...
##  $ smoothness_se          : num  0.0064 0.00522 0.00615 0.00911 0.01149 ...
##  $ compactness_se         : num  0.049 0.0131 0.0401 0.0746 0.0246 ...
##  $ concavity_se           : num  0.0537 0.0186 0.0383 0.0566 0.0569 ...
##  $ concave.points_se      : num  0.0159 0.0134 0.0206 0.0187 0.0188 ...
##  $ symmetry_se            : num  0.03 0.0139 0.0225 0.0596 0.0176 ...
##  $ fractal_dimension_se   : num  0.00619 0.00353 0.00457 0.00921 0.00511 ...
##  $ radius_worst           : num  25.4 25 23.6 14.9 22.5 ...
##  $ texture_worst          : num  17.3 23.4 25.5 26.5 16.7 ...
##  $ perimeter_worst        : num  184.6 158.8 152.5 98.9 152.2 ...
##  $ area_worst             : num  2019 1956 1709 568 1575 ...
##  $ smoothness_worst       : num  0.162 0.124 0.144 0.21 0.137 ...
##  $ compactness_worst      : num  0.666 0.187 0.424 0.866 0.205 ...
##  $ concavity_worst        : num  0.712 0.242 0.45 0.687 0.4 ...
##  $ concave.points_worst   : num  0.265 0.186 0.243 0.258 0.163 ...
##  $ symmetry_worst         : num  0.46 0.275 0.361 0.664 0.236 ...
##  $ fractal_dimension_worst: num  0.1189 0.089 0.0876 0.173 0.0768 ...
```

Data transformation

```
df_c$diagnosis = ifelse(df_c$diagnosis=="M",1,0)
```

Random Sampling

```
s = sample(nrow(df_c),0.8*nrow(df_c))
df_tr = df_c[s,]
df_test = df_c[-s,]
```

Apply logistic regression algorithm

```
log_mod_c = glm(diagnosis~.,data=df_tr,family = binomial,control = list(maxit=100))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(log_mod_c)
```

```
##
```

```
## Call:
## glm(formula = diagnosis ~ ., family = binomial, data = df_tr,
##     control = list(maxit = 100))
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -7.299e-06  -2.110e-08  -2.110e-08   2.110e-08   7.882e-06
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -6.133e+02  1.640e+07       0        1
## id                       -2.993e-07  4.412e-03       0        1
## radius_mean              -1.131e+03  7.461e+06       0        1
## texture_mean              8.464e-01  9.957e+04       0        1
## perimeter_mean            1.521e+02  9.065e+05       0        1
## area_mean                 2.310e+00  2.863e+04       0        1
## smoothness_mean           6.329e+03  5.103e+07       0        1
## compactness_mean         -8.104e+03  2.801e+07       0        1
## concavity_mean            3.177e+03  4.027e+07       0        1
## concave.points_mean      -1.810e+03  6.915e+07       0        1
## symmetry_mean            -2.128e+03  1.455e+07       0        1
## fractal_dimension_mean    1.803e+04  1.268e+08       0        1
## radius_se                 1.425e+03  1.690e+07       0        1
## texture_se               -1.137e+02  1.366e+06       0        1
## perimeter_se              7.479e+01  2.035e+06       0        1
## area_se                  -1.008e+01  2.065e+05       0        1
## smoothness_se             2.049e+04  2.475e+08       0        1
## compactness_se            9.369e+02  6.613e+07       0        1
## concavity_se             -5.562e+03  2.511e+07       0        1
## concave.points_se         1.424e+04  2.058e+08       0        1
## symmetry_se              -2.082e+04  9.430e+07       0        1
## fractal_dimension_se     -4.246e+04  8.519e+08       0        1
## radius_worst              4.838e+01  2.007e+06       0        1
## texture_worst             1.733e+01  9.581e+04       0        1
## perimeter_worst          -1.536e+01  2.315e+05       0        1
## area_worst                4.828e-01  2.428e+04       0        1
## smoothness_worst         -2.504e+03  4.332e+07       0        1
## compactness_worst        -5.501e+02  6.895e+06       0        1
## concavity_worst           8.605e+02  6.146e+06       0        1
## concave.points_worst      2.121e+03  2.859e+07       0        1
## symmetry_worst            2.905e+03  1.050e+07       0        1
## fractal_dimension_worst -3.142e+03  1.274e+08       0        1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5.9928e+02  on 454  degrees of freedom
## Residual deviance: 4.5119e-10  on 423  degrees of freedom
## AIC: 64
##
## Number of Fisher Scoring iterations: 32
```

Prediction

```
pred=predict(log_mod_c,df_test,type="response")
cancer=ifelse(pred>.5,1,0)
```

Accuracy

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
confusionMatrix(factor(df_test$diagnosis),factor(cancer))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 69  1
##          1  3 41
##
##                Accuracy : 0.9649
##                  95% CI : (0.9126, 0.9904)
##     No Information Rate : 0.6316
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9253
##
##  Mcnemar's Test P-Value : 0.6171
##
##             Sensitivity : 0.9583
##             Specificity : 0.9762
##          Pos Pred Value : 0.9857
##          Neg Pred Value : 0.9318
##              Prevalence : 0.6316
##          Detection Rate : 0.6053
##    Detection Prevalence : 0.6140
##       Balanced Accuracy : 0.9673
##
##        'Positive' Class : 0
##
```

Including Plots

ROC Graph

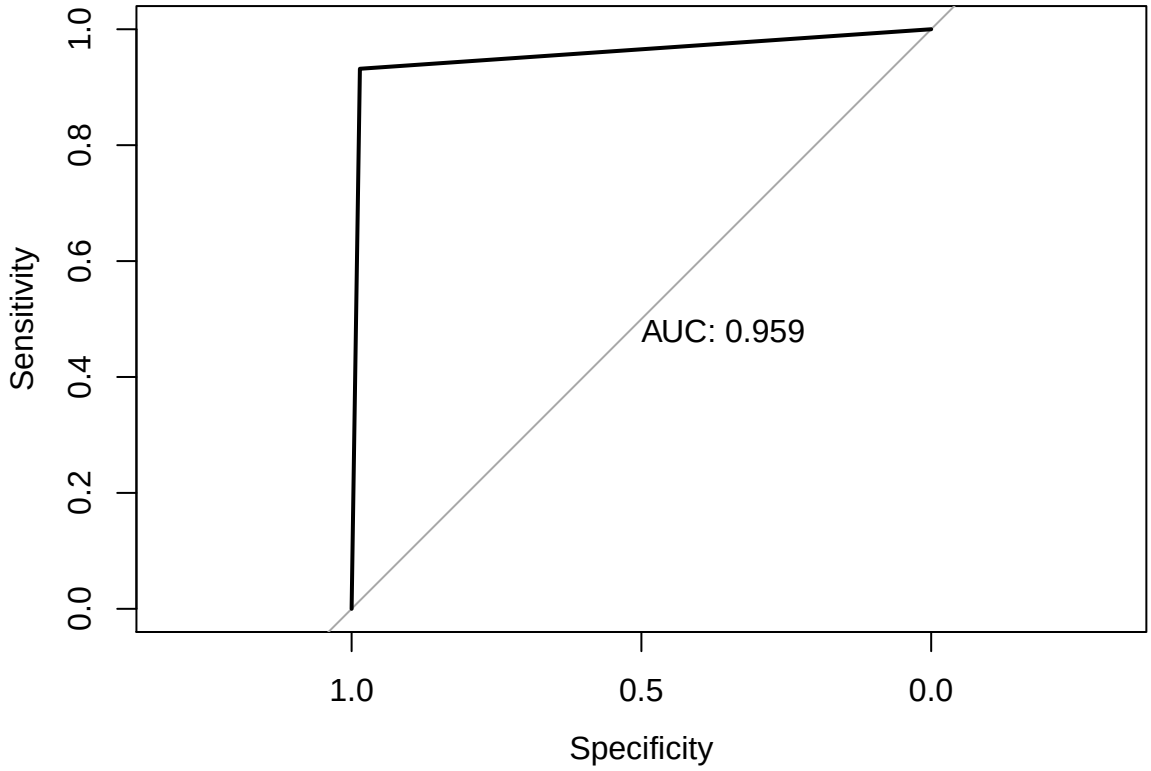
```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```




```
##
## Call:
## roc.default(response = df_test$diagnosis, predictor = cancer,     plot = TRUE, print.auc = TRUE)
##
## Data: cancer in 70 controls (df_test$diagnosis 0) < 44 cases (df_test$diagnosis 1).
## Area under the curve: 0.9588
```