MINI PROJECT - 2

importing necessary libraries

```
import pandas as pd
import numpy as np
import statistics as st
from scipy.stats import norm
from sklearn.datasets import load_wine
```

PROBLEM 1

In a survey conducted by a non-banking financial company, a sample of 200 customers yielded that x of them were highly satisfied with the timely disbursal of their loans.

Write a Python code to perform the following operations:

1.Read an integer input that specifies the number of highly satisfied customers

2.Calculate an approximate 90% confidence interval for the proportion of the loan customers who are highly satisfied with disbursal time

•Find out the Margin of Error using scipy.stats.norm.ppf

•Calculate and print the confidence interval values rounded up to five decimal places and separated by a space

```
#getting an input for highly satisfied customers
proportion=int(input("Give the specifies number of highley satisfied
customers : "))
```

```
Give the specifies number of highley satisfied customers : 172
```

```
#calculation of sample mean
sample_size=200
sample_mean=proportion/sample_size

#Z critical value for 90% confidence interval
z_critical=norm.ppf(0.90)

#calculation of standard error (standard deviation of sample=standard
error)
std_error=np.sqrt((sample_mean * (1 - sample_mean)) / sample_size)

#calculation of margin of error
margin_of_error = z_critical * std_error

#calculation of confience interval
confidence_interval = (sample_mean - margin_of_error).round(5),
(sample_mean + margin_of_error).round(5)
print("The confidence interval is ",confidence_interval)
```

```
The confidence interval is  (0.82856, 0.89144)
```

PROBLEM 2

A radar unit is used to measure the speeds of cars on a motorway. The speeds are normally distributed with a mean of 75 km/hr and a standard deviation of 15 km/hr.

Write a Python code to perform the following operations:

1.Find the probability that a car picked at random is traveling at more than X km/hr

•Take the speed X as an input

•Print the probability value rounded up to four decimal places

```
#getting an input for speed
speed = int(input("Enter the speed : "))
```

```
Enter the speed : 100
```

```
#mean and std are given
sample_mean=75
std=15

# Calculate the probability that a car is traveling at more than X
km/hr
probability = 1 - norm.cdf(speed, sample_mean, std)

# Print the probability value rounded to four decimal places
print(("The probability that a car is traveling at more than"),speed,
("km/hr is"),probability.round(4))
```

```
The probability that a car is traveling at more than 100 km/hr is
0.0478
```

PROBLEM 3

Write a Python program to load the "kerala.csv" data into a DataFrame and perform the following tasks:

1.Explore the DataFrame using info() and describe() functions

2.June and July are the peak months of rainfall. Consider that if it rains more than 500mm, then chances of flood become more; create a Datarame with columns –"YEAR", "JUN_GT_500" (Contains a boolean value to show whether it rained more than 500 mm in the month of June) , "JUL_GT_500" (Contains a boolean value to show whether it rained more than 500 mm in the month of July), and "FLOODS" (Contains a boolean value to show whether it flooded that year)

3.Calculate the probability of flood given it rained more than 500 mm in June (P(A|B))

4.Calculate the probability of rain more than 500 mm in June, given it flooded that year (P(B|A))

5.Probability of flood given it rained more than 500 mm in July

6.Probability of rain more than 500 mm in July given it flooded that year (P(B|A))

```
#load the data
df=pd.read_csv("kerala.csv")
df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 118,\n  \"fields\": [\
n    {\n      \"column\": \"SUBDIVISION\",\n      \"properties\": {\n
\"dtype\": \"category\",\n          \"num_unique_values\": 2,\n
\"samples\": [\n          \"KERALA \",\n          \"KERALA\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"YEAR\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 34,\n        \"min\": 1901,\n        \"max\": 2018,\n
\"num_unique_values\": 118,\n        \"samples\": [\n          1957,\n
1990\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"JAN\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 15.473765773255158,\n        \"min\": 0.0,\n        \"max\":
83.5,\n        \"num_unique_values\": 87,\n        \"samples\": [\n
16.5,\n          28.7\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"FEB\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 16.406290331793723,\n        \"min\": 0.0,\n        \"max\":
79.0,\n        \"num_unique_values\": 96,\n        \"samples\": [\n
57.8,\n          9.1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"MAR\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 30.063861684276155,\n        \"min\": 0.1,\n        \"max\":
217.2,\n        \"num_unique_values\": 108,\n        \"samples\": [\n
0.9,\n          18.2\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"APR\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 44.633452075586625,\n        \"min\": 13.1,\n        \"max\":
238.0,\n        \"num_unique_values\": 116,\n        \"samples\": [\n
66.6,\n          105.9\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"MAY\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 147.5487777364869,\n        \"min\": 53.4,\n        \"max\":
738.8,\n        \"num_unique_values\": 118,\n        \"samples\": [\n
381.2,\n          488.5\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"JUN\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 186.1813630226826,\n        \"min\":
196.8,\n        \"max\": 1098.2,\n        \"num_unique_values\": 116,\
n        \"samples\": [\n          597.9,\n          850.2\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"JUL\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 228.98896571755796,\n        \"min\": 167.5,\n
```

\"max\": 1526.5,\n   \"num_unique_values\": 116,\n
\"samples\": [\n   388.9,\n   520.5\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"AUG\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 181.98046270099286,\n
\"min\": 178.6,\n   \"max\": 1398.9,\n
\"num_unique_values\": 116,\n   \"samples\": [\n   315.3,\
n   293.6\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"SEP\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 121.90113140888297,\n   \"min\": 41.3,\n   \"max\":
526.7,\n   \"num_unique_values\": 117,\n   \"samples\": [\n
110.9,\n   217.2\n   ],\n   \"semantic_type\": \"\",\
n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"OCT\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 93.70525271175732,\n   \"min\":
68.5,\n   \"max\": 567.9,\n   \"num_unique_values\": 116,\n
\"samples\": [\n   165.5,\n   383.5\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"NOV\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 83.20048465495992,\n
\"min\": 31.5,\n   \"max\": 365.6,\n
\"num_unique_values\": 115,\n   \"samples\": [\n   67.7,\n
74.4\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"DEC\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 36.67632969941504,\n   \"min\": 0.1,\n   \"max\":
202.3,\n   \"num_unique_values\": 106,\n   \"samples\": [\n
49.5,\n   87.6\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"
ANNUAL RAINFALL\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 452.16940680740095,\n   \"min\":
2068.8,\n   \"max\": 4473.0,\n   \"num_unique_values\":
118,\n   \"samples\": [\n   3103.3,\n   2693.1\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"FLOODS\",\n   \"properties\":
{\n   \"dtype\": \"category\",\n   \"num_unique_values\":
2,\n   \"samples\": [\n   \"NO\",\n   \"YES\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe","variable_name":"df"}

```
#exploring with info() and describe() functions
df.info()
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 118 entries, 0 to 117
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
```

```
 0   SUBDIVISION       118 non-null    object
 1   YEAR              118 non-null    int64
 2   JAN               118 non-null    float64
 3   FEB               118 non-null    float64
 4   MAR               118 non-null    float64
 5   APR               118 non-null    float64
 6   MAY               118 non-null    float64
 7   JUN               118 non-null    float64
 8   JUL               118 non-null    float64
 9   AUG               118 non-null    float64
 10  SEP               118 non-null    float64
 11  OCT               118 non-null    float64
 12  NOV               118 non-null    float64
 13  DEC               118 non-null    float64
 14   ANNUAL RAINFALL  118 non-null    float64
 15  FLOODS            118 non-null    object
dtypes: float64(13), int64(1), object(2)
memory usage: 14.9+ KB
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 8,\n  \"fields\": [\n{\n      \"column\": \"YEAR\",\n      \"properties\": {\n\"dtype\": \"number\",\n        \"std\": 872.831008812267,\n\"min\": 34.20769893849434,\n        \"max\": 2018.0,\n\"num_unique_values\": 7,\n        \"samples\": [\n          118.0,\n1959.5,\n          1988.75\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n\"column\": \"JAN\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 43.92154638367241,\n        \"min\": 0.0,\n        \"max\": 118.0,\n        \"num_unique_values\": 8,\n\"samples\": [\n          12.218644067796612,\n          5.8,\n118.0\n        ],\n        \"semantic_type\": \"\",\n\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"FEB\",\n      \"properties\": {\n        \"dtype\": \"number\",\n\"std\": 42.34209776194982,\n        \"min\": 0.0,\n        \"max\": 118.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n15.633898305084744,\n          8.350000000000001,\n          118.0\n],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n}\n    },\n    {\n      \"column\": \"MAR\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 71.63385428121957,\n\"min\": 0.1,\n        \"max\": 217.2,\n        \"num_unique_values\": 8,\n        \"samples\": [\n          36.67033898305084,\n28.4,\n          118.0\n        ],\n        \"semantic_type\": \"\",\n\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"APR\",\n      \"properties\": {\n        \"dtype\": \"number\",\n\"std\": 67.53873573082107,\n        \"min\": 13.1,\n        \"max\": 238.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n110.33050847457629,\n          110.4,\n          118.0\n        ],\n\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"MAY\",\n      \"properties\": {\n\"dtype\": \"number\",\n        \"std\": 215.02367626043295,\n
```

```
\"min\": 53.4,\n        \"max\": 738.8,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
228.6449152542373,\n          184.60000000000002,\n          118.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"JUN\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 340.50698201935796,\
n        \"min\": 118.0,\n        \"max\": 1098.2,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
651.6177966101693,\n          625.5999999999999,\n          118.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"JUL\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 461.6678955792624,\n
\"min\": 118.0,\n        \"max\": 1526.5,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
698.220338983051,\n          691.65,\n          118.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"AUG\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 410.54971891172914,\n
\"min\": 118.0,\n        \"max\": 1398.9,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
430.3694915254237,\n          386.25,\n          118.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"SEP\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 153.06319968696016,\n
\"min\": 41.3,\n        \"max\": 526.7,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
246.20762711864407,\n          223.55,\n          118.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"OCT\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 165.06734241120245,\n
\"min\": 68.5,\n        \"max\": 567.9,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
293.20762711864404,\n          284.3,\n          118.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"NOV\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 102.82392367699929,\n
\"min\": 31.5,\n        \"max\": 365.6,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
162.3110169491526,\n          152.45,\n          118.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"DEC\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 67.04074739646785,\n
\"min\": 0.1,\n        \"max\": 202.3,\n        \"num_unique_values\":
8,\n        \"samples\": [\n          40.00932203389831,\n
31.1,\n          118.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"
ANNUAL RAINFALL\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1443.4353893200296,\n        \"min\":
118.0,\n        \"max\": 4473.0,\n        \"num_unique_values\": 8,\n
```

\"samples\": [\n   2925.4050847457625,\n   2934.3,\n
118.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
JUN_GT_500=(df['JUN']>500).astype('int64')
JUL_GT_500=(df['JUL']>500).astype('int64')
FLOODS=df['FLOODS'].replace("YES",1).replace("NO",0)
df=pd.DataFrame({'YEAR':df['YEAR'],'JUN_GT_500':JUN_GT_500,'JUL_GT_500
':JUL_GT_500,'FLOODS':FLOODS})
df
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 118,\n \"fields\": [\
n   {\n   \"column\": \"YEAR\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 34,\n   \"min\": 1901,\n
\"max\": 2018,\n   \"num_unique_values\": 118,\n
\"samples\": [\n   1957,\n   1990,\n   1905\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"JUN_GT_500\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0,\n   \"min\": 0,\n   \"max\": 1,\n
\"num_unique_values\": 2,\n   \"samples\": [\n   0,\n
1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"JUL_GT_500\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0,\n   \"min\": 0,\n
\"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\":
[\n   0,\n   1\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"FLOODS\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0,\n   \"min\": 0,\n
\"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\":
[\n   0,\n   1\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"df"}

```
data=pd.crosstab(df['FLOODS'],df['JUN_GT_500'],margins=True,margins_na
me='total')
data
```

{"summary":"{\n \"name\": \"data\",\n \"rows\": 3,\n \"fields\": [\
n   {\n   \"column\": \"FLOODS\",\n   \"properties\": {\n
\"dtype\": \"string\",\n   \"num_unique_values\": 3,\n
\"samples\": [\n   0,\n   1,\n   \"total\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": 0,\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 9,\n   \"min\": 6,\n
\"max\": 25,\n   \"num_unique_values\": 3,\n   \"samples\":
[\n   19,\n   6,\n   25\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": 1,\n   \"properties\": {\n

\"dtype\": \"number\",\n   \"std\": 27,\n   \"min\": 39,\n
\"max\": 93,\n   \"num_unique_values\": 3,\n   \"samples\":
[\n   39,\n   54,\n   93\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"total\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 34,\n
\"min\": 58,\n   \"max\": 118,\n   \"num_unique_values\":
3,\n   \"samples\": [\n   58,\n   60,\n
118\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"data"}

```
data1=pd.crosstab(df['FLOODS'],df['JUL_GT_500'],margins=True,margins_n
ame='total')
data1
```

{"summary":"{\n \"name\": \"data1\",\n \"rows\": 3,\n \"fields\":
[\n   {\n   \"column\": \"FLOODS\",\n   \"properties\": {\n
\"dtype\": \"string\",\n   \"num_unique_values\": 3,\n
\"samples\": [\n   0,\n   1,\n   \"total\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": 0,\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 10,\n   \"min\": 3,\n
\"max\": 22,\n   \"num_unique_values\": 3,\n   \"samples\":
[\n   19,\n   3,\n   22\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": 1,\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 29,\n   \"min\": 39,\n
\"max\": 96,\n   \"num_unique_values\": 3,\n   \"samples\":
[\n   39,\n   57,\n   96\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"total\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 34,\n
\"min\": 58,\n   \"max\": 118,\n   \"num_unique_values\":
3,\n   \"samples\": [\n   58,\n   60,\n
118\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"data1"}

```
s=118
p_f=round(60/s,4)

#Calculate the probability of flood given it rained more than 500 mm
in June (P(A|B))
p_jn=round(93/s,4)
p_f_and_jn=round(54/s,4)
p_f_jn=round(p_f_and_jn/p_jn,4)
print("The probability of flood given it rained more than 500 mm in
June is ",p_f_jn)
```

```
#Calculate the probability of rain more than 500 mm in June, given it flooded that year (P(B|A))
p_jn_f=round(p_f_jn*p_jn/p_f,4)  #p(B/A)=(P(A/B)*P(B))/P(A)
print("The probability of rain more than 500 mm in June, given it flooded that year is ",p_jn_f)

#Probability of flood given it rained more than 500 mm in July
p_jl=round(96/s,4)
p_f_and_jl=round(57/s,4)
p_jl_f=round(p_f_and_jl/p_jl,4)
print("The probability of flood given it rained more than 500 mm in July is ",p_jl_f)

#Probability of rain more than 500 mm in July given it flooded that year (P(B|A)
p_jn_jl=round(p_jl_f*p_jl/p_f,4)  #p(B/A)=(P(A/B)*P(B))/P(A)
print("The probability of rain more than 500 mm in July given it flooded that year is ",p_jn_jl)
```

```
The probability of flood given it rained more than 500 mm in June is 0.5806
The probability of rain more than 500 mm in June, given it flooded that year is  0.8998
The probability of flood given it rained more than 500 mm in July is 0.5938
The probability of rain more than 500 mm in July given it flooded that year is  0.9501
```

** PROBLEM 4**

Write a Python program to load the wine dataset using the Sklearn library to a DataFrame and perform the following tasks:

1.Convert the dataset into DataFrame using pandas.

2.Generate the sample size of 50 and give a random state as 100.

3.Calculate Z-critical, Margin of Error, and Confidence Interval for alcohol at 95% significance interval on generated sample data.

```
#load the dataset
wine=load_wine()

#convert the dataset into dataframe
df=pd.DataFrame(wine.data,columns=wine['feature_names'])
df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 178,\n  \"fields\": [\n    {\n      \"column\": \"alcohol\",\n      \"properties\": {\n \"dtype\": \"number\",\n        \"std\": 0.8118265380058575,\n \"min\": 11.03,\n        \"max\": 14.83,\n
```

```
\"num_unique_values\": 126,\n        \"samples\": [\n          11.62,\
n          13.64,\n          13.69\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"malic_acid\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.1171460976144627,\n        \"min\": 0.74,\n        \"max\": 5.8,\n
\"num_unique_values\": 133,\n        \"samples\": [\n          1.21,\n
2.83,\n          1.8\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"ash\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.27434400906081485,\n        \"min\": 1.36,\n
\"max\": 3.23,\n        \"num_unique_values\": 79,\n
\"samples\": [\n          2.31,\n          2.43,\n          2.52\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"alcalinity_of_ash\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3.339563767173505,\n        \"min\": 10.6,\n        \"max\": 30.0,\n
\"num_unique_values\": 63,\n        \"samples\": [\n          25.5,\n
28.5,\n          15.6\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"magnesium\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 14.282483515295665,\n        \"min\":
70.0,\n        \"max\": 162.0,\n        \"num_unique_values\": 53,\n
\"samples\": [\n          126.0,\n          85.0,\n          162.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"total_phenols\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.6258510488339893,\n        \"min\": 0.98,\n        \"max\": 3.88,\n
\"num_unique_values\": 97,\n        \"samples\": [\n          1.68,\n
2.11,\n          1.35\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"flavanoids\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.9988586850169467,\n        \"min\":
0.34,\n        \"max\": 5.08,\n        \"num_unique_values\": 132,\n
\"samples\": [\n          3.18,\n          2.5,\n          3.17\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"nonflavanoid_phenols\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.12445334029667937,\n        \"min\": 0.13,\n        \"max\": 0.66,\n
\"num_unique_values\": 39,\n        \"samples\": [\n          0.58,\n
0.41,\n          0.39\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"proanthocyanins\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.5723588626747613,\n        \"min\":
0.41,\n        \"max\": 3.58,\n        \"num_unique_values\": 101,\n
\"samples\": [\n          0.75,\n          1.77,\n          1.42\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"color_intensity\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
```

2.318285871822413,\n   \"min\": 1.28,\n   \"max\": 13.0,\n
\"num_unique_values\": 132,\n   \"samples\": [\n   2.95,\n
3.3,\n   5.1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"hue\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0.22857156582982338,\n   \"min\": 0.48,\n
\"max\": 1.71,\n   \"num_unique_values\": 78,\n
\"samples\": [\n   1.22,\n   1.04,\n   1.45\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"od280/od315_of_diluted_wines\",\
n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0.7099904287650504,\n   \"min\": 1.27,\n   \"max\":
4.0,\n   \"num_unique_values\": 122,\n   \"samples\": [\n
4.0,\n   1.82,\n   1.59\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"proline\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\":
314.9074742768491,\n   \"min\": 278.0,\n   \"max\": 1680.0,\
n   \"num_unique_values\": 121,\n   \"samples\": [\n
1375.0,\n   1270.0,\n   735.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe","variable_name":"df"}

```
#generate the sample size of 50 and giving a random state as 100
sample=df.sample(50,random_state=100)
sample.head()
```

{"summary":"{\n \"name\": \"sample\",\n \"rows\": 50,\n \"fields\":
[\n   {\n   \"column\": \"alcohol\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 0.7686663775657162,\n
\"min\": 11.64,\n   \"max\": 14.38,\n
\"num_unique_values\": 45,\n   \"samples\": [\n   12.25,\n
11.66,\n   12.17\n   ],\n   \"semantic_type\": \"\",\
n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"malic_acid\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 1.1124258471044812,\n
\"min\": 1.09,\n   \"max\": 5.04,\n   \"num_unique_values\":
46,\n   \"samples\": [\n   3.37,\n   1.88,\n
1.39\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"ash\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0.24324204792253956,\n   \"min\": 1.75,\n
\"max\": 2.86,\n   \"num_unique_values\": 33,\n
\"samples\": [\n   2.75,\n   2.61,\n   2.48\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"alcalinity_of_ash\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
3.0556097624546497,\n   \"min\": 11.2,\n   \"max\": 25.5,\n
\"num_unique_values\": 28,\n   \"samples\": [\n   17.5,\n
23.0,\n   19.0\n   ],\n   \"semantic_type\": \"\",\n

```
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"magnesium\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 17.36053464802585,\n        \"min\":
80.0,\n        \"max\": 162.0,\n        \"num_unique_values\": 29,\n
\"samples\": [\n          116.0,\n          123.0,\n          112.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"total_phenols\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.6147629043553527,\n        \"min\": 0.98,\n        \"max\": 3.38,\n
\"num_unique_values\": 38,\n        \"samples\": [\n          1.8,\n
3.0,\n          2.8\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"flavanoids\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.9205948609237311,\n        \"min\":
0.34,\n        \"max\": 3.29,\n        \"num_unique_values\": 45,\n
\"samples\": [\n          0.66,\n          2.76,\n          2.29\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"nonflavanoid_phenols\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.12110460986524108,\n        \"min\": 0.13,\n        \"max\": 0.6,\n
\"num_unique_values\": 24,\n        \"samples\": [\n          0.32,\n
0.42,\n          0.48\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"proanthocyanins\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.6027337720752007,\n        \"min\":
0.55,\n        \"max\": 3.28,\n        \"num_unique_values\": 45,\n
\"samples\": [\n          0.97,\n          1.03,\n          0.94\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"color_intensity\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2.7598655726580015,\n        \"min\": 1.28,\n        \"max\": 13.0,\n
\"num_unique_values\": 47,\n        \"samples\": [\n          2.9,\n
4.7,\n          3.8\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"hue\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.25421676432781953,\n        \"min\": 0.48,\n
\"max\": 1.45,\n        \"num_unique_values\": 35,\n
\"samples\": [\n          0.58,\n          0.6,\n          0.54\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"od280/od315_of_diluted_wines\",\
n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.7179006809824755,\n        \"min\": 1.27,\n        \"max\":
3.71,\n        \"num_unique_values\": 44,\n        \"samples\": [\n
3.59,\n          3.38,\n          3.4\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"proline\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
285.49208216619917,\n        \"min\": 325.0,\n        \"max\":
1515.0,\n        \"num_unique_values\": 44,\n        \"samples\": [\n
```

```
685.0,\n          1050.0,\n          428.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"sample"}

#sample size is given
sample_size=50

#calculation of sample mean
sample_mean=sample['alcohol'].mean()

#calculation of z-critical
z_critical=norm.ppf(0.95)

#calculation of standard error
std=sample['alcohol'].std()
std_error=std/np.sqrt(sample_size)

#calculation of margin of error
margin_of_error=z_critical*std_error

#calculation of confidence interval
confidence_interval=(sample_mean-margin_of_error).round(5),
(sample_mean+margin_of_error).round(5)

print("The z_critical is ",z_critical.round(5),"\nThe margin or error
is ",margin_of_error.round(5),"\nThe confidence interval is
",confidence_interval)
```

```
The z_critical is  1.64485
The margin or error is  0.17881
The confidence interval is  (12.79479, 13.15241)
```