```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
d=pd.read_csv('/content/Credit Card Customer Data.csv')
d
```

	Sl_No	Customer Key	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Tot
0	1	87073	100000	2	1	1	
1	2	38414	50000	3	0	10	
2	3	17341	50000	7	1	3	
3	4	40496	30000	5	1	1	
4	5	47437	100000	6	0	12	
...	...	...	...	...	...	...	
655	656	51108	99000	10	1	10	
656	657	60732	84000	10	1	13	
657	658	53834	145000	8	1	9	
658	659	80655	172000	10	1	15	
659	660	80150	167000	9	0	12	

660 rows × 7 columns

```
d.dtypes
```

```
Sl_No                 int64
Customer Key          int64
Avg_Credit_Limit      int64
Total_Credit_Cards    int64
Total_visits_bank     int64
Total_visits_online   int64
Total_calls_made      int64
dtype: object
```

```
#check for null values
d.isna().sum()
```

```
Sl_No                 0
Customer Key          0
Avg_Credit_Limit      0
Total_Credit_Cards    0
Total_visits_bank     0
Total_visits_online   0
Total_calls_made      0
dtype: int64
```

```
#heatmap for null values
sns.heatmap(d.isnull(), cmap='plasma')
```

```
<Axes: >
```


```
df=d.drop(['Sl_No'], axis=1)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 660 entries, 0 to 659
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Customer Key         660 non-null    int64
 1   Avg_Credit_Limit     660 non-null    int64
 2   Total_Credit_Cards   660 non-null    int64
 3   Total_visits_bank    660 non-null    int64
 4   Total_visits_online  660 non-null    int64
 5   Total_calls_made     660 non-null    int64
dtypes: int64(6)
memory usage: 31.1 KB
```

```
df.describe()
```

	Customer Key	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Tota
count	660.000000	660.000000	660.000000	660.000000	660.000000	
mean	55141.443939	34574.242424	4.706061	2.403030	2.606061	
std	25627.772200	37625.487804	2.167835	1.631813	2.935724	
min	11265.000000	3000.000000	1.000000	0.000000	0.000000	
25%	33825.250000	10000.000000	3.000000	1.000000	1.000000	
50%	53874.500000	18000.000000	5.000000	2.000000	2.000000	
75%	77202.500000	48000.000000	6.000000	4.000000	4.000000	
max	99843.000000	200000.000000	10.000000	5.000000	15.000000	

```
#all columns
cols=df.columns
cols
```

```
Index(['Customer Key', 'Avg_Credit_Limit', 'Total_Credit_Cards',
       'Total_visits_bank', 'Total_visits_online', 'Total_calls_made'],
      dtype='object')
```

```
#to find the value counts of each column
for cols in df.columns:
  print('Value count of', cols, 'is', '\n', df[cols].value_counts())
```

```
            ..
43000      1
146000     1
155000     1
200000     1
167000     1
Name: Avg_Credit_Limit, Length: 110, dtype: int64
Value count of Total_Credit_Cards is
```

```
Value count of Total_visits_online is
 2     189
0     144
1     109
4      69
5      54
3      44
15     10
7       7
10      6
12      6
8       6
11      5
13      5
9       4
14      1
6       1
Name: Total_visits_online, dtype: int64
Value count of Total_calls_made is
 4     108
0      97
2      91
1      90
3      83
6      39
7      35
9      32
8      30
5      29
10     26
Name: Total calls made, dtype: int64
```

```
#to find unique values in each column
for cols in df.columns:
  print('Unique values of', cols, 'is', '\n', df[cols].unique())
```

```
33317 99596 72430 16676 40486 90958 67212 44226 94251 61776 55275 18609
54477 12122 28208 68003 79632 73811 72892 51773 96163 61234 55849 56156
54838 35254 46635 97825 83125 35483 15129 83290 56486 31903 45909 14263
46813 81878 35549 85799 39122 81531 69965 18595 44398 32352 40898 27101
33457 45088 23302 27408 65372 21531 56843 17165 89328 20072 71402 47496
24808 17036 67193 34423 97109 55382 51811 53936 66504 53207 18514 51319
36340 36934 95925 49771 22919 21233 74544 52025 45652 73952 49418 77026
49331 75775 54906 94666 11698 34677 95610 41380 38033 85337 38994 67911
92956 77641 57565 53814 30712 19785 31384 16374 50878 78002 83459 91987
51552 72156 24998 45673 11596 87485 28414 81863 33240 11466 23881 44645
49844 92782 22824 26767 26678 50412 17933 34495 22610 41159 64672 62483
85614 96548 19137 69028 70779 38244 67046 64897 46223 36628 17565 77381
11799 81940 66706 87838 94437 33790 44402 29886 66804 47866 61996 15318
89635 71681 71862 96186 22348 36243 88807 82376 98126 80347 17649 62807
92522 57459 44579 45476 61994 11398 24702 27824 45878 72431 19215 23409
16418 85122 55060 55478 65574 31113 96929 78912 68439 62864 31515 77954
88207 78618 31551 75792 29864 45440 97954 90189 55090 17703 33991 88884
45808 50706 92140 88123 53932 65908 25321 87456 48602 97530 48657 76209
49913 53002 61122 82807 93496 64519 31950 23110 96297 28408 41287 52460
26604 58019 87219 36839 12663 48667 42887 14439 60851 41266 37438 65747
81166 20570 14816 11265 24980 37934 70707 84351 89446 17325 64774 53166
45341 94595 55170 92489 92933 36504 40508 15798 70101 77613 84360 48402
46776 67258 44804 29919 65781 12456 62649 74446 36632 76024 75065 51682
18397 29102 56367 95147 44379 76957 42921 23102 61324 49690 20043 44144
53552 62530 41741 22842 65825 77826 61216 83192 82023 73000 64550 90131
17382 27117 94529 21717 81910 76492 43000 48692 27476 15086 43034 99131
13140 99437 91242 39285 63710 90860 35585 58708 57451 69868 43679 30256
26334 47848 17377 39644 29176 55706 51771 83585 51867 68040 75417 34775
85645 83545 44157 38125 75398 90999 70376 33295 80942 26493 97850 43841
79885 59316 83466 81510 35268 11734 88411 96269 87683 26063 42479 58116
67282 84888 75366 14377 59074 96534 31870 24748 68920 67637 60839 59170
90586 56270 87670 47703 35421 58511 76398 93310 36836 46373 94700 67860
99473 68862 93381 46548 74083 48660 13720 72339 99284 47198 67415 44403
58276 85234 31948 90191 49341 11562 16253 80623 94391 50598 40019 77910
89832 98216 54495 47650 32107 84192 53916 32584 97285 20337 15585 20620
75009 76203 33837 14916 16180 49493 70974 40217 88442 17538 90839 99843
```

```
Unique values of Total_visits_online is
 [ 1 10  3 12 11  2  5  4  0 14  7 13 15  6  8  9]
Unique values of Total_calls_made is
 [ 0  9  4  3  8  2  1  7  5  6 10]
```

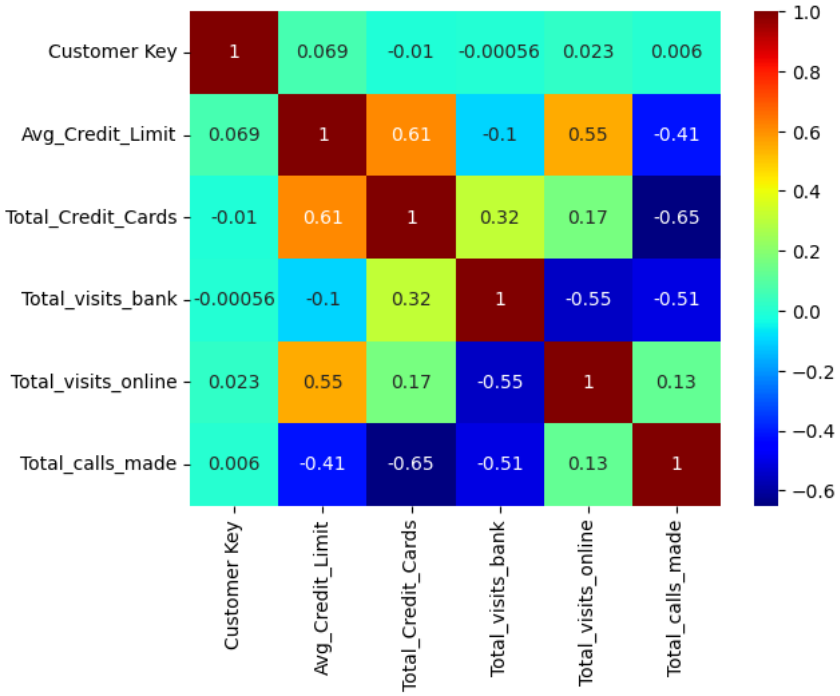
Visualization

```
corr=df.corr()
corr
```

	Customer Key	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_onli
Customer Key	1.000000	0.068604	-0.010281	-0.000560	0.0225
Avg_Credit_Limit	0.068604	1.000000	0.608860	-0.100312	0.5513
Total_Credit_Cards	-0.010281	0.608860	1.000000	0.315796	0.1677
Total_visits_bank	-0.000560	-0.100312	0.315796	1.000000	-0.5518
Total_visits_online	0.022506	0.551385	0.167758	-0.551861	1.0000
Total_calls_made	0.005968	-0.414352	-0.651251	-0.506016	0.1272

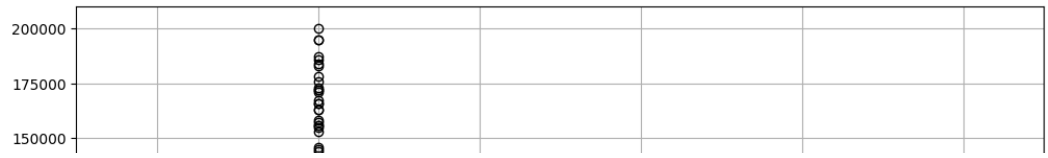
```
sns.heatmap(corr, cmap='jet', annot=True)
```

```
<Axes: >
```



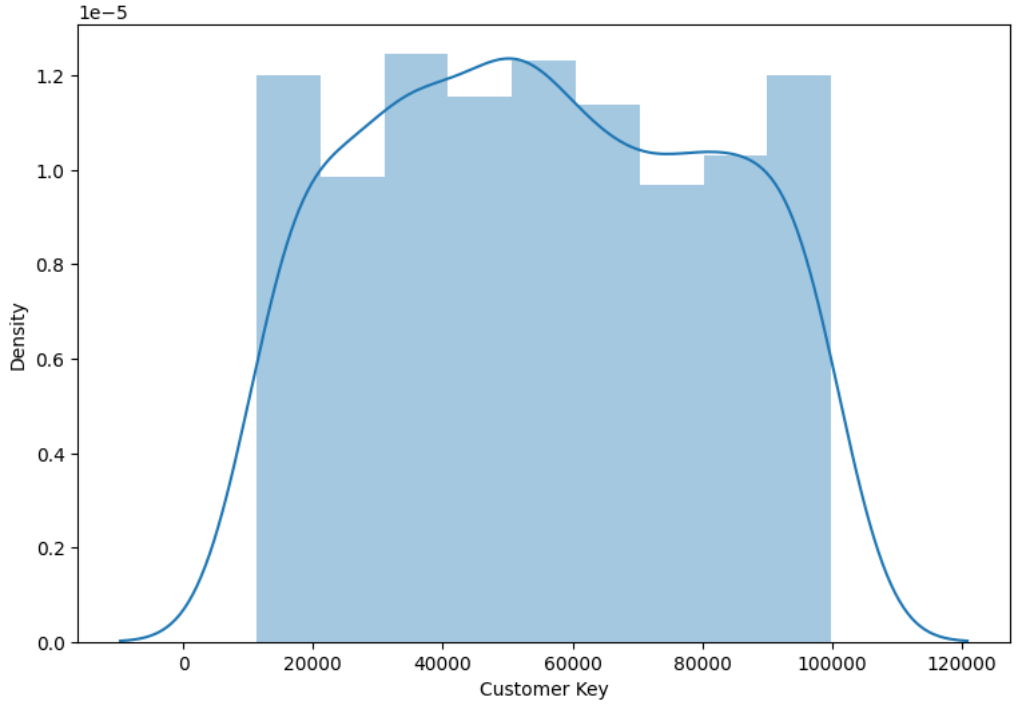

```
#boxplot for cols 'Customer Key', 'Avg_Credit_Limit', 'Total_Credit_Cards','Total_visits_bank', 'Total_visits_online', 'Total_calls_made'
plt.figure(figsize=(12,6))
df.boxplot(column=['Customer Key', 'Avg_Credit_Limit', 'Total_Credit_Cards',
                   'Total_visits_bank', 'Total_visits_online', 'Total_calls_made'])
```

```
<Axes: >
```

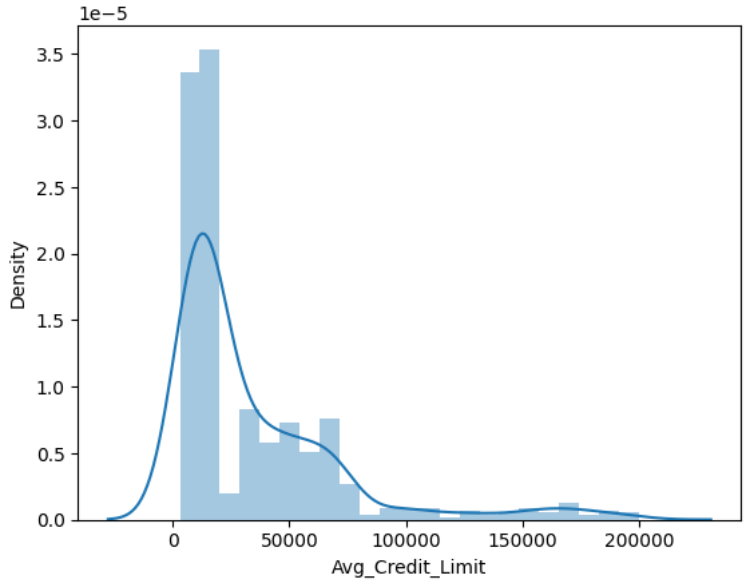



```
#to show the distplot of each column
plt.figure(figsize=(9,6))
for cols in df.columns:
  sns.distplot(df[cols], kde=True)
  plt.show()
```

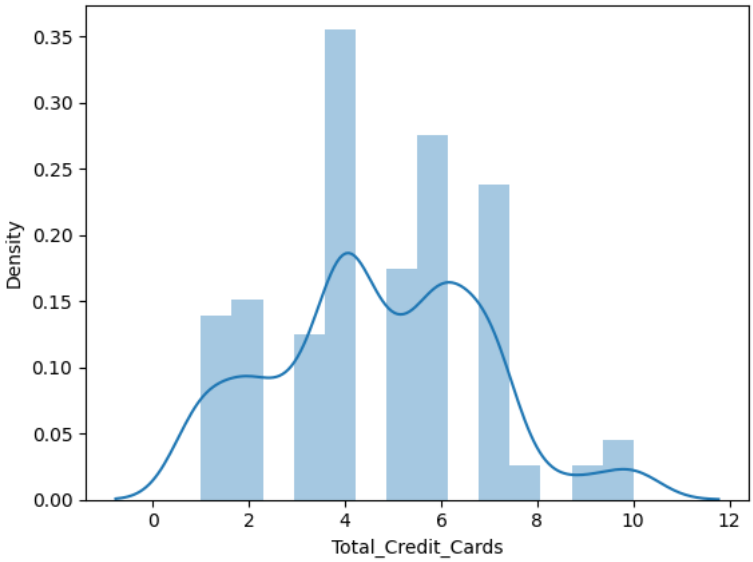
```
<ipython-input-318-07d5cb09d209>:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[cols], kde=True)
```



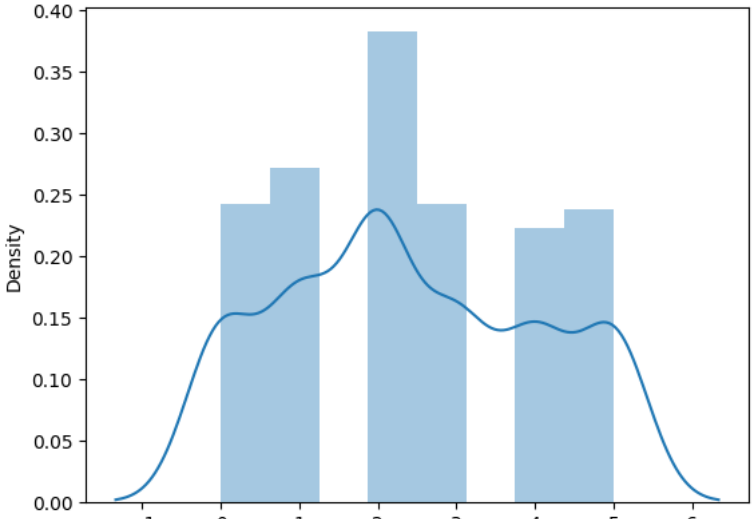

```
<ipython-input-318-07d5cb09d209>:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[cols], kde=True)
```




```
<ipython-input-318-07d5cb09d209>:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[cols], kde=True)
```




```
<ipython-input-318-07d5cb09d209>:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[cols], kde=True)
```



Clustering

```
<ipython-input-318-07d5cb09d209>:4: UserWarning:
```

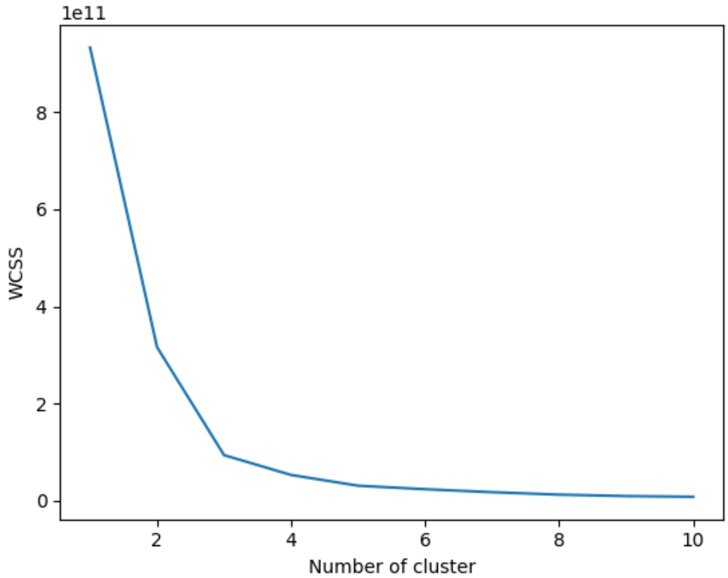
```
X_clus=df.drop('Customer Key',axis=1)
X_clus
```

```
#import  Kmeans
from sklearn.cluster import KMeans
```

```
#to the optimal cluster we can use elbow point/method
wcss=[]
for i in range(1,11):
  km=KMeans(n_clusters=i,init='k-means++',random_state=42)
  km.fit(X_clus)
  wcss.append(km.inertia_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
```

```
import matplotlib.pyplot as plt
plt.plot(range(1,11),wcss)
plt.xlabel("Number of cluster")
plt.ylabel("WCSS")
plt.show()
#3 is the elbow point or 5 is the optimal cluster number
```




```
# KMeans is a clustering algorithm in scikit-learn used for K-means clustering
from sklearn.cluster import KMeans
km=KMeans(n_clusters=3,init='k-means++',random_state=42)
km.fit(X_clus)
y_pred=km.predict(X_clus)
y_pred
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
array([0, 0, 0, 2, 0, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
```

```
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0,
       0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 2, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0, 0, 2, 0, 0, 2, 0, 2,
       2, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0, 2, 0, 0, 0, 0, 0, 2, 0,
       2, 0, 0, 2, 2, 0, 0, 0, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0,
       2, 2, 0, 2, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 0,
       0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0,
       0, 0, 0, 0, 2, 0, 0, 0, 0, 2, 0, 0, 2, 0, 2, 0, 0, 2, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 2, 0, 0, 2, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1],
      dtype=int32)
```

```
X_clus['Clusters']=y_pred
y_pred
```

```
array([0, 0, 0, 2, 0, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0,
       0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 2, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0, 0, 2, 0, 0, 2, 0, 2,
       2, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0, 2, 0, 0, 0, 0, 0, 2, 0,
       2, 0, 0, 2, 2, 0, 0, 0, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0,
       2, 2, 0, 2, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 0,
       0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0,
       0, 0, 0, 0, 2, 0, 0, 0, 0, 2, 0, 0, 2, 0, 2, 0, 0, 2, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 2, 0, 0, 2, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1],
      dtype=int32)
```

```
X_clus
```

	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_made	Cl
0	100000	2	1	1	0	
1	50000	3	0	10	9	
2	50000	7	1	3	4	
3	30000	5	1	1	4	
4	100000	6	0	12	3	
...	...	...	...	...	...	
655	99000	10	1	10	0	
656	84000	10	1	13	2	
657	145000	8	1	9	1	
658	172000	10	1	15	0	
659	167000	9	0	12	2	

660 rows × 6 columns

```
X_clus['Clusters'].value_counts()
```

```
2    431
0    190
1     39
Name: Clusters, dtype: int64
```

```
cluster_analysis=X_clus.groupby('Clusters').mean()
print(cluster_analysis)
```

```
            Avg_Credit_Limit  Total_Credit_Cards  Total_visits_bank  \
Clusters
0              57031.578947            5.731579           3.210526
1             154205.128205            8.743590           0.589744
2              13849.187935            3.888631           2.211137

            Total_visits_online  Total_calls_made
Clusters
0                      1.568421          2.052632
1                     10.871795          1.000000
2                      2.315545          4.491879
```

```
dfx=X_clus
dfx
```

	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_made	Cl
0	100000	2	1	1	0	
1	50000	3	0	10	9	
2	50000	7	1	3	4	
3	30000	5	1	1	4	
4	100000	6	0	12	3	
...	...	...	...	...	...	
655	99000	10	1	10	0	
656	84000	10	1	13	2	
657	145000	8	1	9	1	
658	172000	10	1	15	0	
659	167000	9	0	12	2	

660 rows × 6 columns

```
dfx.rename(columns={'Clusters':'Spenders'}, inplace=True)
dfx
```

	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_made	Sp
0	100000	2	1	1	0	
1	50000	3	0	10	9	
2	50000	7	1	3	4	
3	30000	5	1	1	4	
4	100000	6	0	12	3	
...	...	...	...	...	...	
655	99000	10	1	10	0	
656	84000	10	1	13	2	
657	145000	8	1	9	1	
658	172000	10	1	15	0	
659	167000	9	0	12	2	

660 rows × 6 columns

```
dfx['Spenders'].replace({0:'Med spender',1:'High spender',2:'Low spender'}, inplace=True)
dfx
```

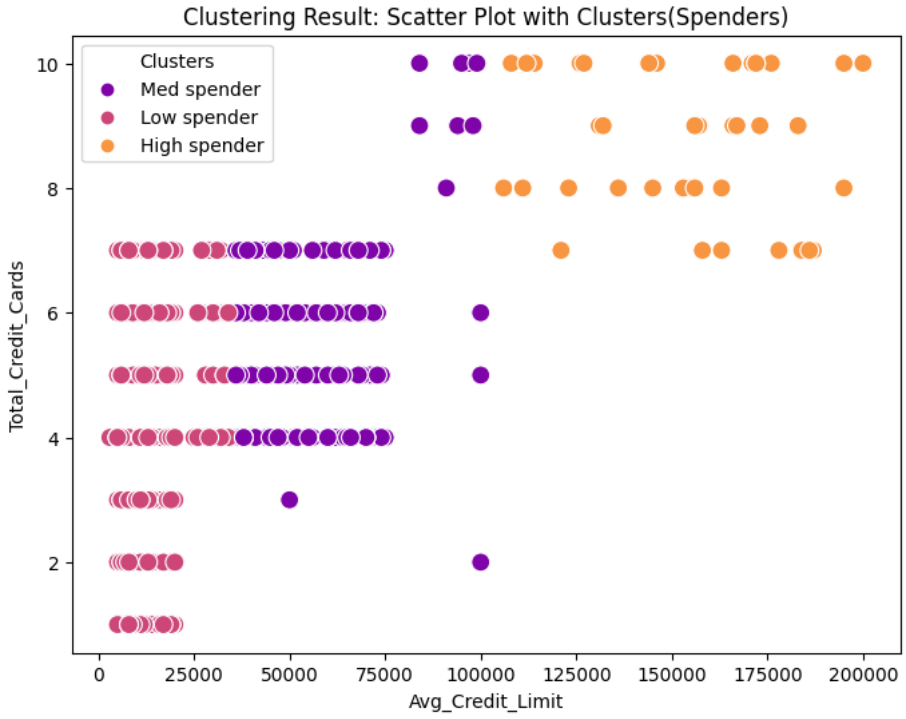
	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_made	Sp
0	100000	2	1	1	0	
1	50000	3	0	10	9	
2	50000	7	1	3	4	
3	30000	5	1	1	4	
4	100000	6	0	12	3	

```
dfx["Spenders"].value_counts()
```

```
Low spender     431
Med spender     190
High spender     39
Name: Spenders, dtype: int64
```

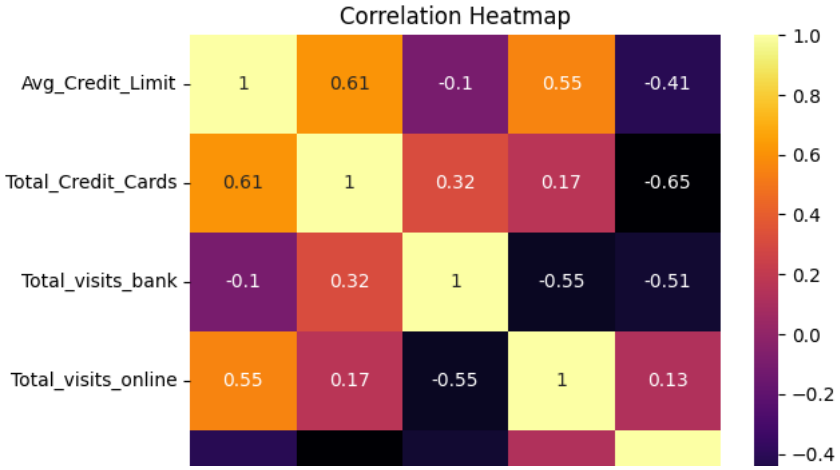
Visualization of new data (dfx)

```
#to plot visualize the clusters(Spenders) using 'Avg_Credit_Limit' and 'Total_Credit_Cards'
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Avg_Credit_Limit', y='Total_Credit_Cards', hue='Spenders', data=dfx, palette='plasma', s=100)
plt.xlabel('Avg_Credit_Limit')
plt.ylabel('Total_Credit_Cards')
plt.title('Clustering Result: Scatter Plot with Clusters(Spenders)')
plt.legend(title='Clusters', loc='upper left')
plt.show()
```

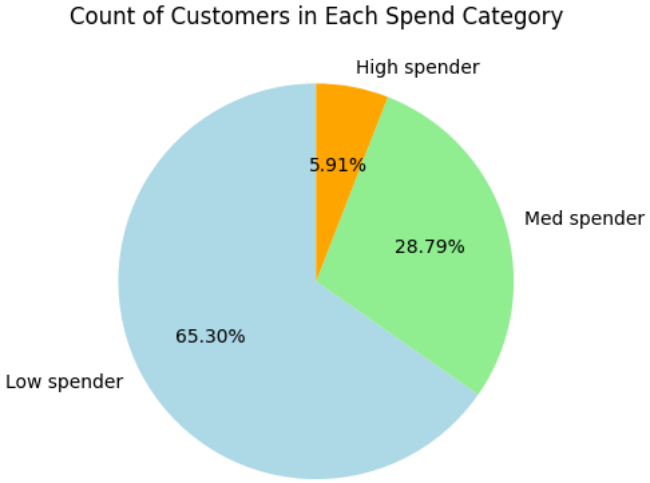



```
nw_corr=dfx.corr()
sns.heatmap(nw_corr, annot=True, cmap='inferno')
plt.title('Correlation Heatmap')
plt.show()
```

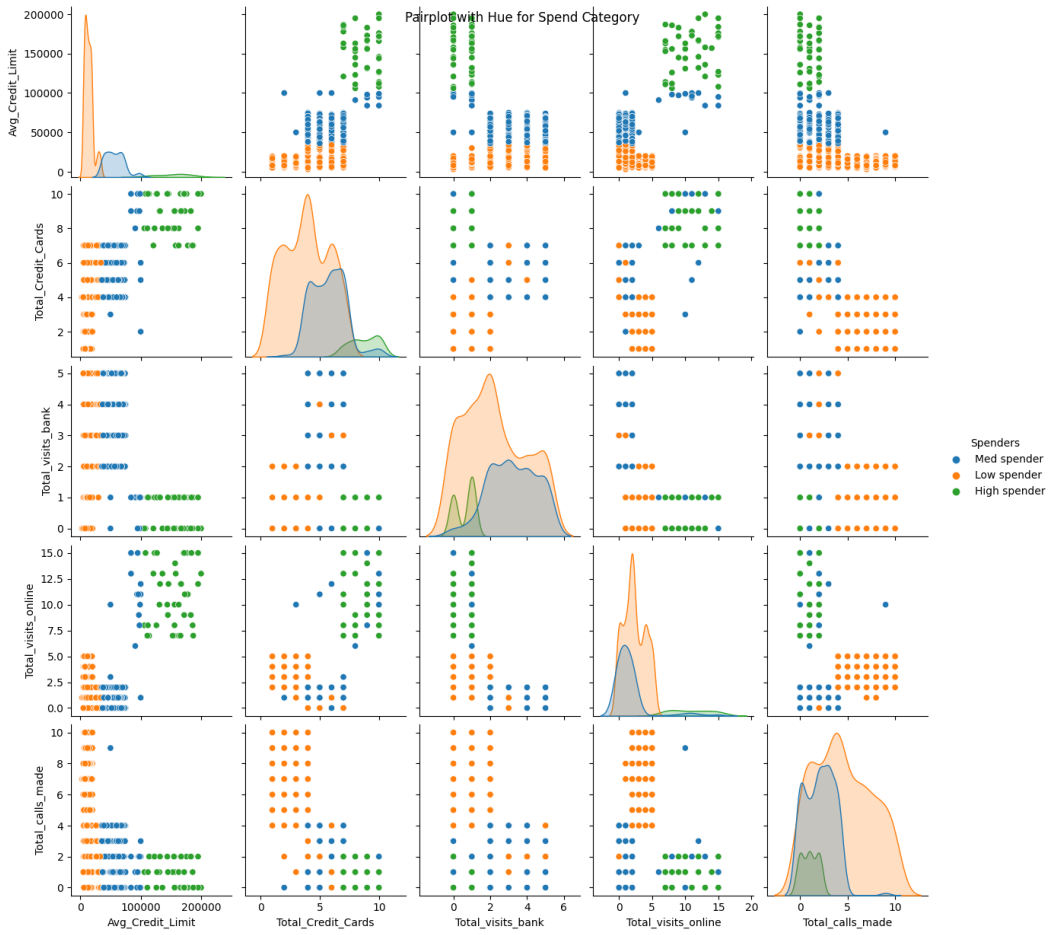
```
<ipython-input-333-033cea9e1b2d>:1: FutureWarning: The default value of numeric_only in DataFrame.corr
  nw_corr=dfx.corr()
```



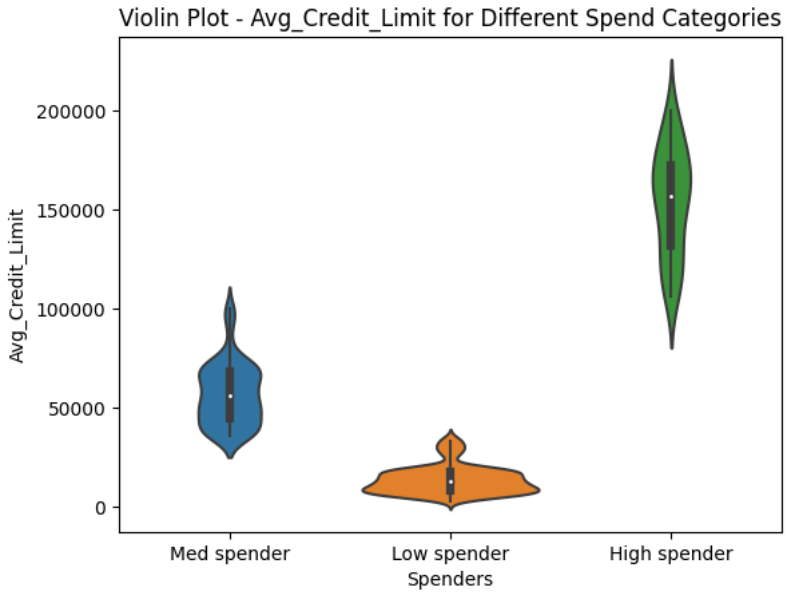

```
#to plot pie chart for different type spenders
spenders_count=dfx["Spenders"].value_counts()
plt.pie(spenders_count, labels=spenders_count.index, autopct='%1.2f%%', startangle=90, colors=['lightblue', 'lightgreen', 'orange'])
plt.title('Count of Customers in Each Spend Category')
plt.show()
```



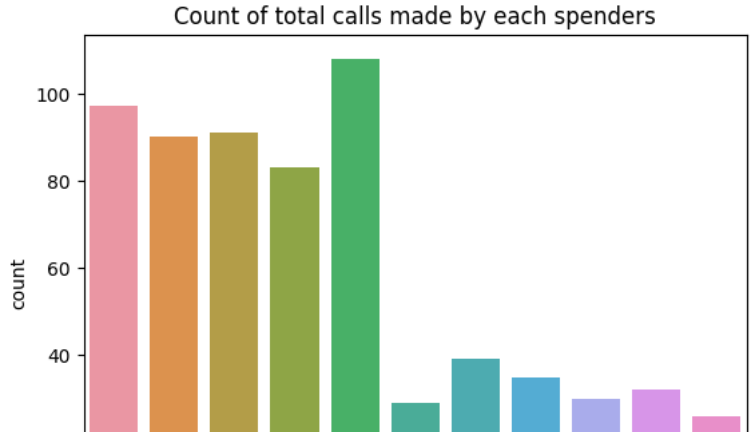

```
#to plot featuers by target columns
sns.pairplot(dfx, hue='Spenders', diag_kind='kde')
plt.suptitle('Pairplot with Hue for Spend Category')
plt.show()
```



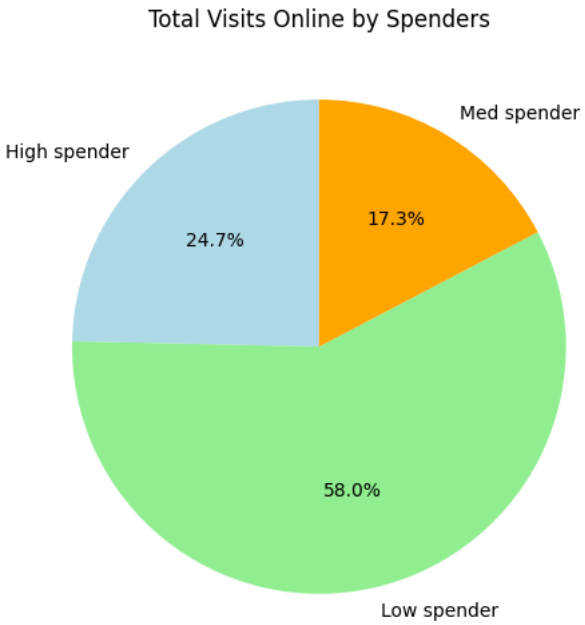

```
#to plot spenders by avg credit limit
sns.violinplot(x='Spenders', y='Avg_Credit_Limit', data=dfx)
plt.title('Violin Plot - Avg_Credit_Limit for Different Spend Categories')
plt.show()
```




```
#count of total calls made by each spenders
sns.countplot(x='Total_calls_made', data=dfx)
plt.title('Count of total calls made by each spenders')
plt.show()
```




```
#total visits online by spenders
visits_by_spenders=dfx.groupby('Spenders')['Total_visits_online'].sum()
plt.figure(figsize=(8, 6))
plt.pie(visits_by_spenders, labels=visits_by_spenders.index, autopct='%1.1f%%', startangle=90, colors=['lightblue', 'lightgreen', 'orange
plt.title('Total Visits Online by Spenders')
plt.show()
```



Predciction

```
dfx.dtypes
```

```
Avg_Credit_Limit        int64
Total_Credit_Cards      int64
Total_visits_bank       int64
Total_visits_online     int64
Total_calls_made        int64
Spenders               object
dtype: object
```

```
#convert the catagorical values in spenders back to numerical values
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
dfx['Spenders']=encoder.fit_transform(dfx['Spenders'])
dfx
```

	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_made	Sp
0	100000	2	1	1	0	
1	50000	3	0	10	9	
2	50000	7	1	3	4	
3	30000	5	1	1	4	
4	100000	6	0	12	3	
...	...	...	...	...	...	

```
X=dfx.drop(columns=['Spenders'], axis=1)
y=dfx['Spenders']
```

```
X.shape
```

```
(660, 5)
```

660 rows × 6 columns

```
y.shape
```

```
(660,)
```

```
#split the dataset
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.2, random_state=42)
X_train.shape
```

```
(528, 5)
```

```
X_test.shape
```

```
(132, 5)
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.fit_transform(X_test)
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, ConfusionMatrixDisplay
```

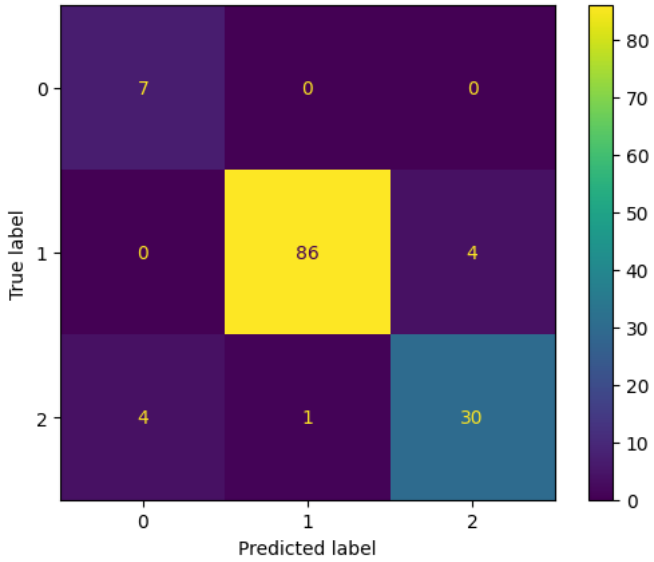
```
classifiers = [
    ('KNeighborsClassifier', KNeighborsClassifier()),
    ('DecisionTreeClassifier', DecisionTreeClassifier()),
    ('RandomForestClassifier', RandomForestClassifier()),
    ('GradientBoostingClassifier', GradientBoostingClassifier()),
    ('AdaBoostClassifier', AdaBoostClassifier()),
    ('SVC', SVC()),
    ('GaussianNB', GaussianNB())
]
```

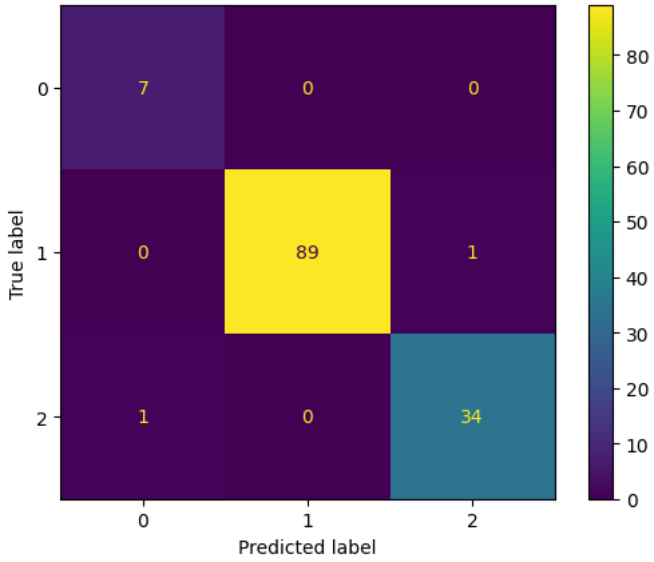
```
result=pd.DataFrame(columns=['Classifier', 'Accuracy'])
```

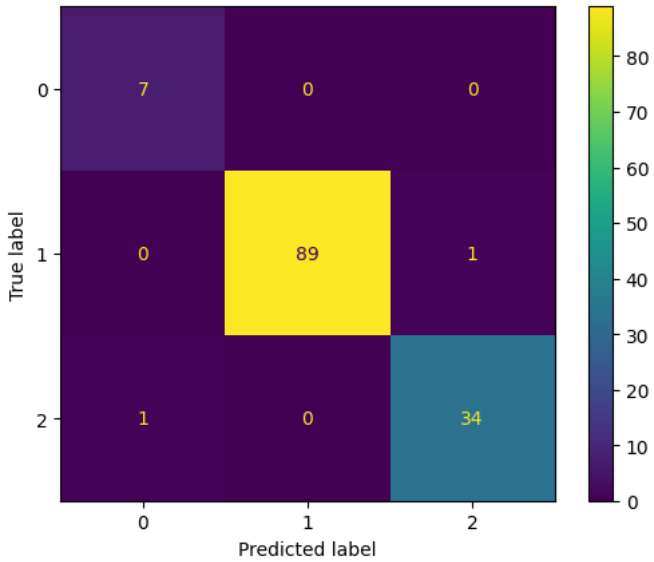
```
for clf_name,clf in classifiers:
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy=accuracy_score(y_test, y_pred)
    report=classification_report(y_test,y_pred)
    cmatrix=ConfusionMatrixDisplay.from_predictions(y_test,y_pred)
    print("Accuracy is:",accuracy)
    print("Classification report is:",report)
    print("Confusion matrix is:",cmatrix)
    result=result.append({'Classifier': clf_name, 'Accuracy': accuracy}, ignore_index=True)
```

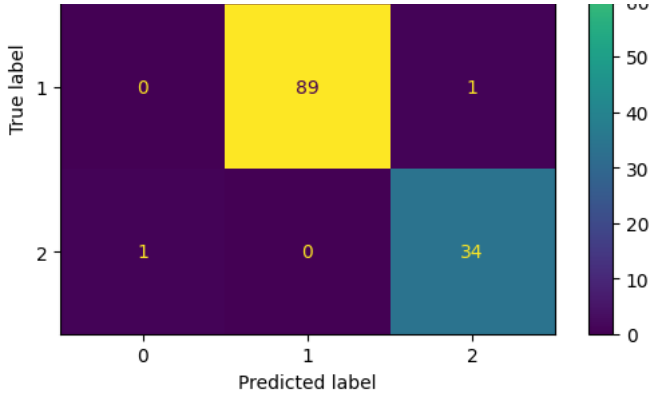
```
Accuracy is: 0.9318181818181818
Classification report is:               precision    recall  f1-score   support

           0       0.64      1.00      0.78         7
           1       0.99      0.96      0.97        90
           2       0.88      0.86      0.87        35

    accuracy                           0.93       132
   macro avg       0.84      0.94      0.87       132
weighted avg       0.94      0.93      0.93       132

Confusion matrix is: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7eb14ec
Accuracy is: 0.9848484848484849
Classification report is:               precision    recall  f1-score   support

           0       0.88      1.00      0.93         7
           1       1.00      0.99      0.99        90
           2       0.97      0.97      0.97        35

    accuracy                           0.98       132
   macro avg       0.95      0.99      0.97       132
weighted avg       0.99      0.98      0.99       132

Confusion matrix is: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7eb14e2
<ipython-input-350-e92d88ea8e19>:10: FutureWarning: The frame.append method is deprecated and will be r
  result=result.append({'Classifier': clf_name, 'Accuracy': accuracy}, ignore_index=True)
<ipython-input-350-e92d88ea8e19>:10: FutureWarning: The frame.append method is deprecated and will be r
  result=result.append({'Classifier': clf_name, 'Accuracy': accuracy}, ignore_index=True)
Accuracy is: 0.9848484848484849
Classification report is:               precision    recall  f1-score   support

           0       0.88      1.00      0.93         7
           1       1.00      0.99      0.99        90
           2       0.97      0.97      0.97        35

    accuracy                           0.98       132
   macro avg       0.95      0.99      0.97       132
weighted avg       0.99      0.98      0.99       132

Confusion matrix is: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7eb14c3
<ipython-input-350-e92d88ea8e19>:10: FutureWarning: The frame.append method is deprecated and will be r
  result=result.append({'Classifier': clf_name, 'Accuracy': accuracy}, ignore_index=True)
Accuracy is: 0.9848484848484849
Classification report is:               precision    recall  f1-score   support

           0       0.88      1.00      0.93         7
           1       1.00      0.99      0.99        90
           2       0.97      0.97      0.97        35

    accuracy                           0.98       132
   macro avg       0.95      0.99      0.97       132
weighted avg       0.99      0.98      0.99       132

Confusion matrix is: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7eb14c5
Accuracy is: 0.9848484848484849
Classification report is:               precision    recall  f1-score   support

           0       0.88      1.00      0.93         7
           1       1.00      0.99      0.99        90
           2       0.97      0.97      0.97        35

    accuracy                           0.98       132
   macro avg       0.95      0.99      0.97       132
weighted avg       0.99      0.98      0.99       132

Confusion matrix is: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7eb14be
<ipython-input-350-e92d88ea8e19>:10: FutureWarning: The frame.append method is deprecated and will be r
  result=result.append({'Classifier': clf_name, 'Accuracy': accuracy}, ignore_index=True)
<ipython-input-350-e92d88ea8e19>:10: FutureWarning: The frame.append method is deprecated and will be r
  result=result.append({'Classifier': clf_name, 'Accuracy': accuracy}, ignore_index=True)
Accuracy is: 0.9545454545454546
Classification report is:               precision    recall  f1-score   support

           0       0.67      0.86      0.75         7
           1       1.00      0.98      0.99        90
           2       0.91      0.91      0.91        35

    accuracy                           0.95       132
   macro avg       0.86      0.92      0.88       132
weighted avg       0.96      0.95      0.96       132

Confusion matrix is: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7eb14f1
Accuracy is: 0.9242424242424242
Classification report is:               precision    recall  f1-score   support

           0       0.58      1.00      0.74         7
           1       1.00      0.94      0.97        90
           2       0.86      0.86      0.86        35

    accuracy                           0.92       132
```

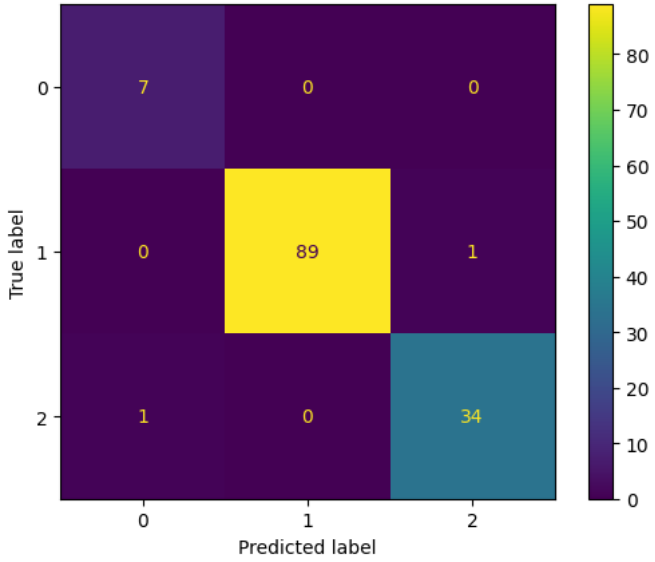
```
   macro avg       0.81      0.93      0.86       132
weighted avg       0.94      0.92      0.93       132

Confusion matrix is: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7eb14bc
<ipython-input-350-e92d88ea8e19>:10: FutureWarning: The frame.append method is deprecated and will be r
  result=result.append({'Classifier': clf_name, 'Accuracy': accuracy}, ignore_index=True)
<ipython-input-350-e92d88ea8e19>:10: FutureWarning: The frame.append method is deprecated and will be r
  result=result.append({'Classifier': clf_name, 'Accuracy': accuracy}, ignore_index=True)
```

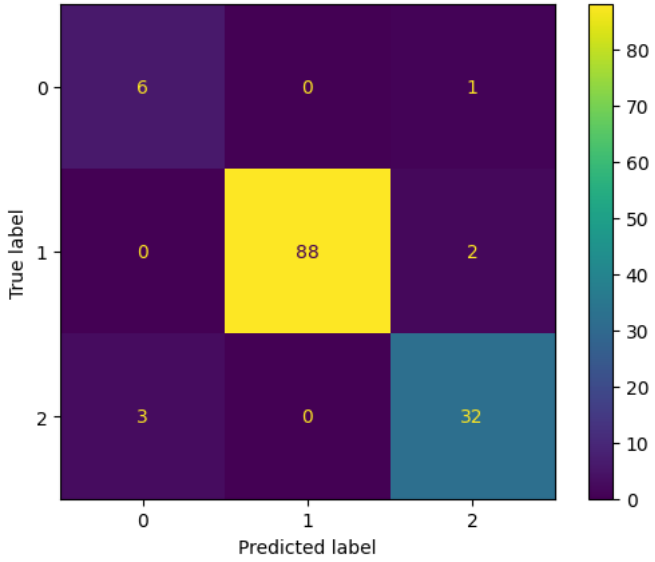


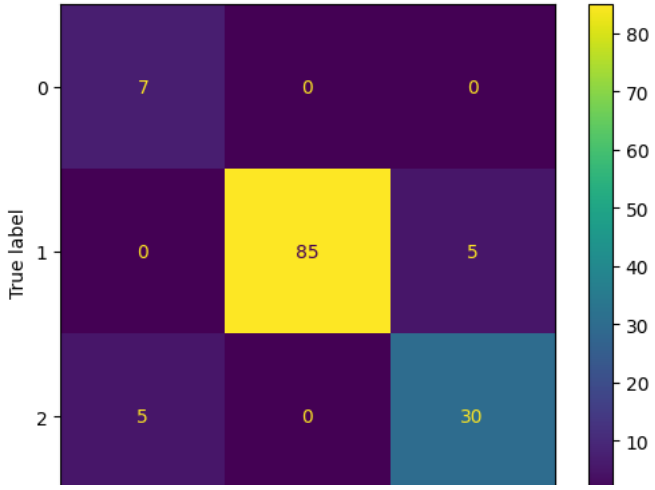





True label
Predicted label
0
89
1
1
0
34
0
1
2
50
40
30
20
10
0


True label
Predicted label
7
0
0
0
89
1
1
0
34
0
1
2
80
70
60
50
40
30
20
10
0


True label
Predicted label
6
0
1
0
88
2
3
0
32
0
1
2
80
70
60
50
40
30
20
10
0


True label
7
0
0
0
85
5
5
0
30
0
1
2
80
70
60
50
40
30
20
10

```
print(result)
```

```
                    Classifier  Accuracy
0        KNeighborsClassifier  0.931818
1      DecisionTreeClassifier  0.984848
2      RandomForestClassifier  0.984848
3  GradientBoostingClassifier  0.984848
4          AdaBoostClassifier  0.984848
5                         SVC  0.954545
6                  GaussianNB  0.924242
```

```
#to find the model with the highest accuracy
highest_accuracy_model=result.loc[result['Accuracy'].idxmax()]
print("\nModel with highest accuracy:")
print(highest_accuracy_model)
```

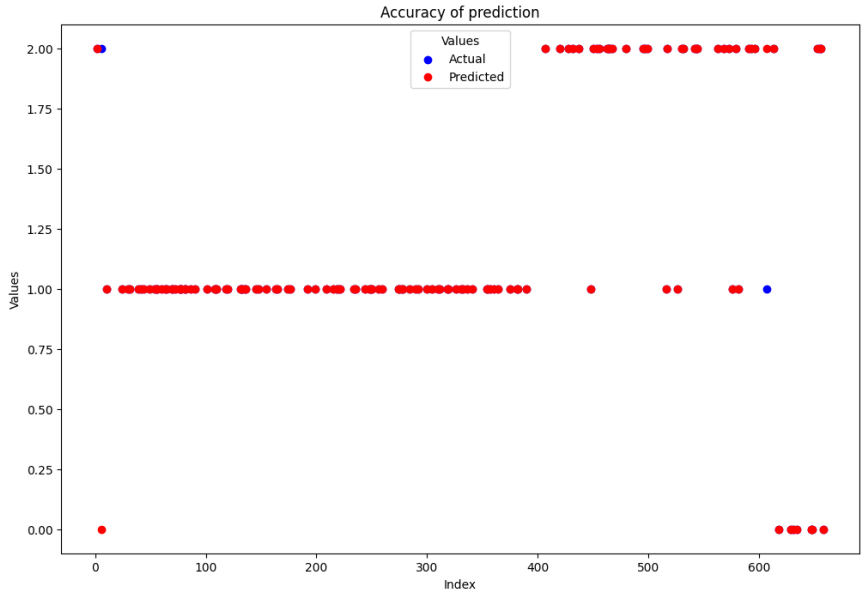
```
Model with highest accuracy:
Classifier    DecisionTreeClassifier
Accuracy                    0.984848
Name: 1, dtype: object
```

```
dtc=DecisionTreeClassifier().fit(X_train,y_train)
y_pred_dtc=dtc.predict(X_test)
ohcheck=pd.DataFrame()
ohcheck['Actual']=y_test
ohcheck['Predicted']=y_pred_dtc
ohcheck['Difference']=y_test-y_pred_dtc
ohcheck.sort_index()
```

	Actual	Predicted	Difference
2	2	2	0
6	2	0	2
10	1	1	0
24	1	1	0
30	1	1	0
...	...	...	...
648	0	0	0
653	2	2	0
655	2	2	0
656	2	2	0
658	0	0	0

132 rows × 3 columns

```
#to plot the accuracy of prediction
plt.figure(figsize=(12,8))
plt.scatter(ohcheck.index, ohcheck['Actual'], color='blue')
plt.scatter(ohcheck.index, ohcheck['Predicted'], color='red')
plt.title('Accuracy of prediction')
plt.xlabel('Index')
plt.ylabel('Values')
plt.legend(title='Values', loc=9, labels=['Actual', 'Predicted'])
plt.show()
```



✓ 1s completed at 12:14 PM

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.