Importing the libraries

In [1]:
```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

Loading the dataset

In [2]:
```
data = pd.read_csv('Bank-data (4).csv')
```

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

Out[3]:

	Index	interest_rate	credit	Gender	previous	duration	Churn
0	0	1.334	0	1	0	117	no
1	1	0.767	0	0	1	274	yes
2	2	4.858	0	1	0	167	no
3	3	4.120	0	0	0	686	yes
4	4	4.856	0	1	0	159	no

Shape

In [4]:
```
data.shape
```

Out[4]:
```
(518, 7)
```

Columns

In [5]:
```
data.columns
```

Out[5]:
```
Index(['Index', 'interest_rate', 'credit', 'Gender', 'previous', 'duration',
       'Churn'],
      dtype='object')
```

Basic Information

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 518 entries, 0 to 517
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Index          518 non-null    int64  
 1   interest_rate  518 non-null    float64
 2   credit         518 non-null    int64  
 3   Gender         518 non-null    int64  
 4   previous       518 non-null    int64  
 5   duration       518 non-null    int64  
 6   Churn          518 non-null    object 
dtypes: float64(1), int64(5), object(1)
memory usage: 28.5+ KB
```

Statistical summary

In [7]:
```
data.describe()
```

Out[7]:

	Index	interest_rate	credit	Gender	previous	duration
count	518.000000	518.000000	518.000000	518.000000	518.000000	518.000000
mean	258.500000	2.903736	0.034749	0.268340	0.127413	384.252896
std	149.677988	1.878504	0.183321	0.443524	0.333758	343.622003
min	0.000000	0.635000	0.000000	0.000000	0.000000	9.000000
25%	129.250000	1.055500	0.000000	0.000000	0.000000	156.250000
50%	258.500000	1.706500	0.000000	0.000000	0.000000	268.500000
75%	387.750000	4.957000	0.000000	1.000000	0.000000	483.000000
max	517.000000	4.970000	1.000000	1.000000	1.000000	2653.000000

Check for the null values

In [8]:
```
data.isnull().sum()/len(data)*100
```

Out[8]:
```
Index            0.0
interest_rate    0.0
credit           0.0
Gender           0.0
previous         0.0
duration         0.0
Churn            0.0
dtype: float64
```

In [9]:
```
data_dup = data.duplicated().any()
print(data_dup)
```

```
False
```

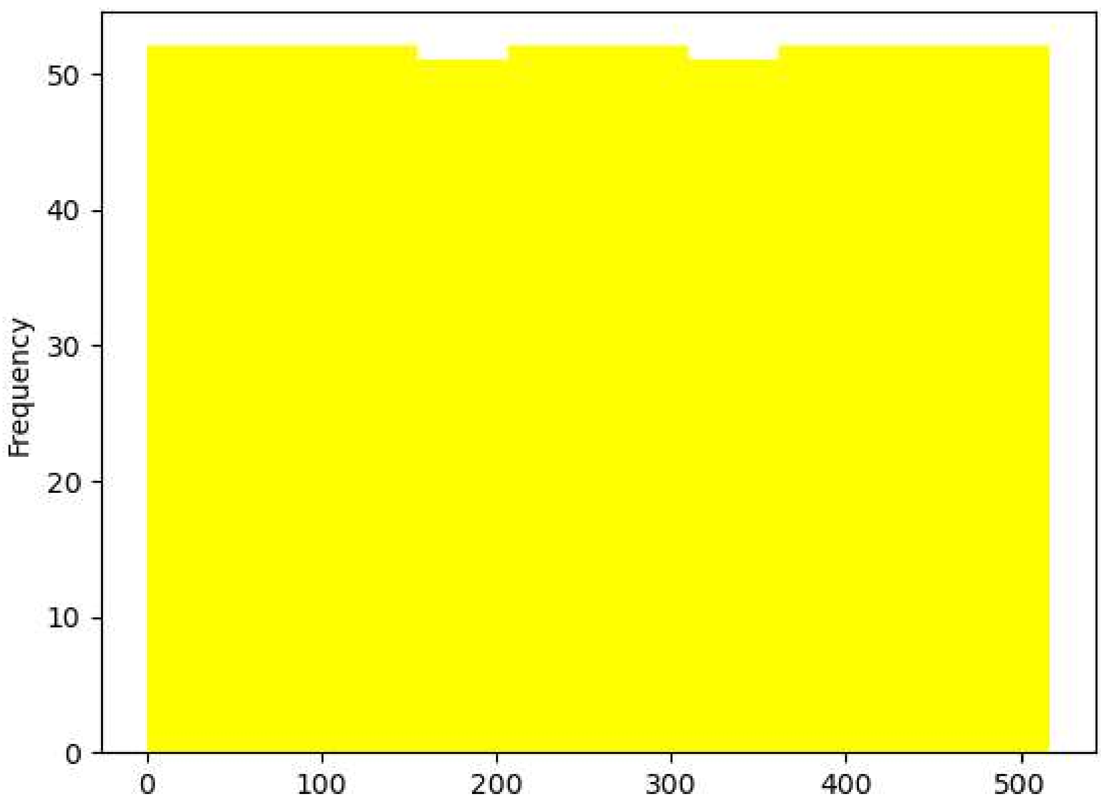
In [10]:
```
data['Churn'].value_counts()
```

Out[10]:
```
no     259
yes    259
Name: Churn, dtype: int64
```

In [11]:
```
data['Churn'] = data['Churn'].astype('category')
data['Churn'] = data['Churn'].cat.codes
```

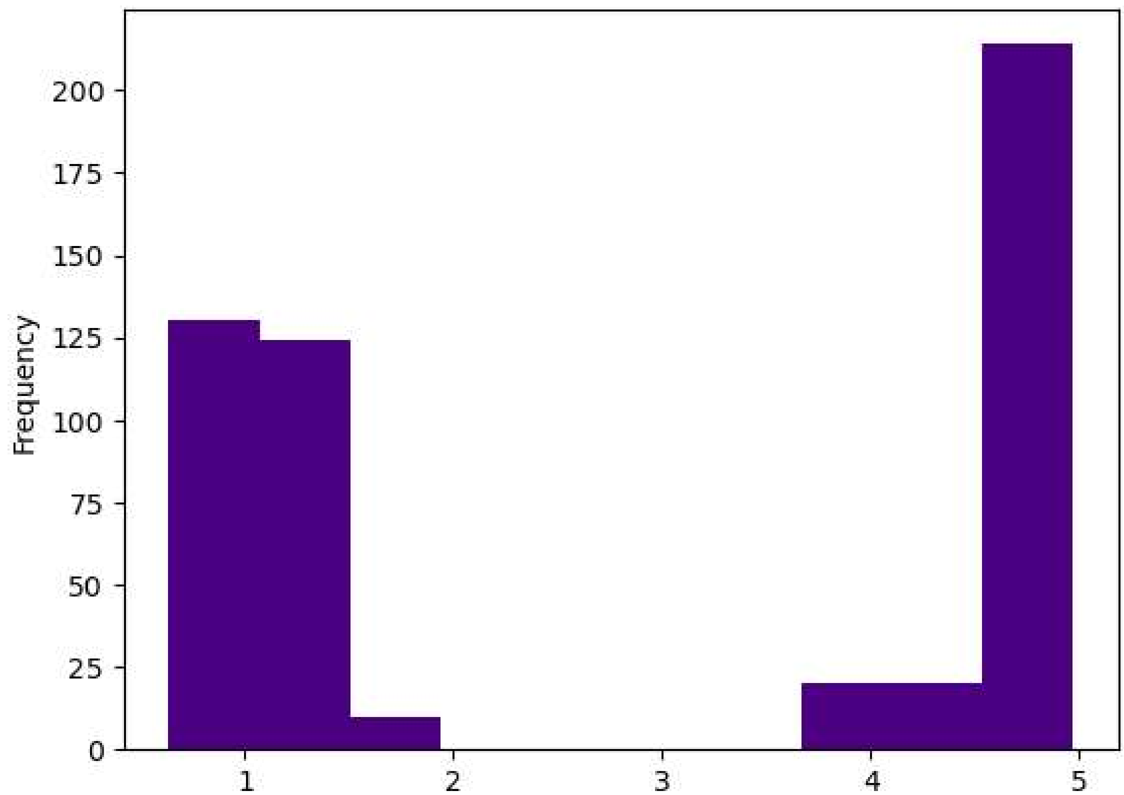
In [12]:
```
data['Index'].plot(kind='hist',color='yellow')
```

Out[12]: <Axes: ylabel='Frequency'>



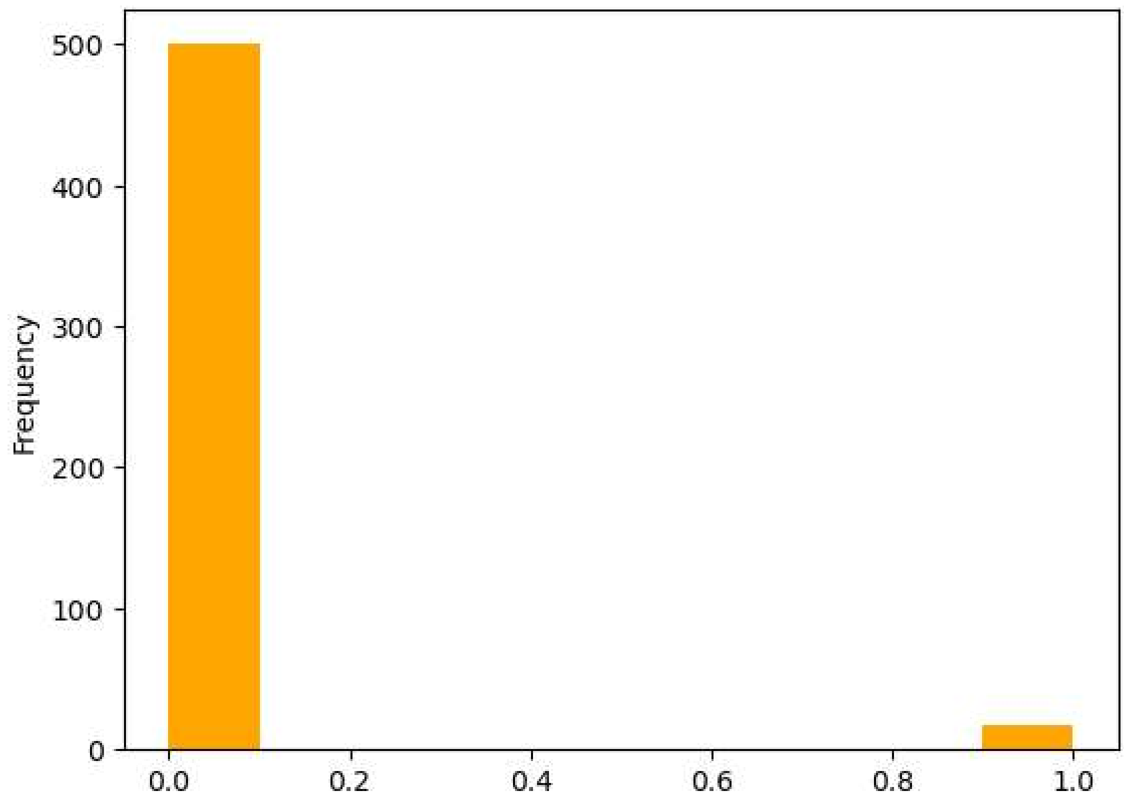
In [13]:
```
data['interest rate'].plot(kind='hist',color='indigo')
```

Out[13]:
```
<Axes: ylabel='Frequency'>
```



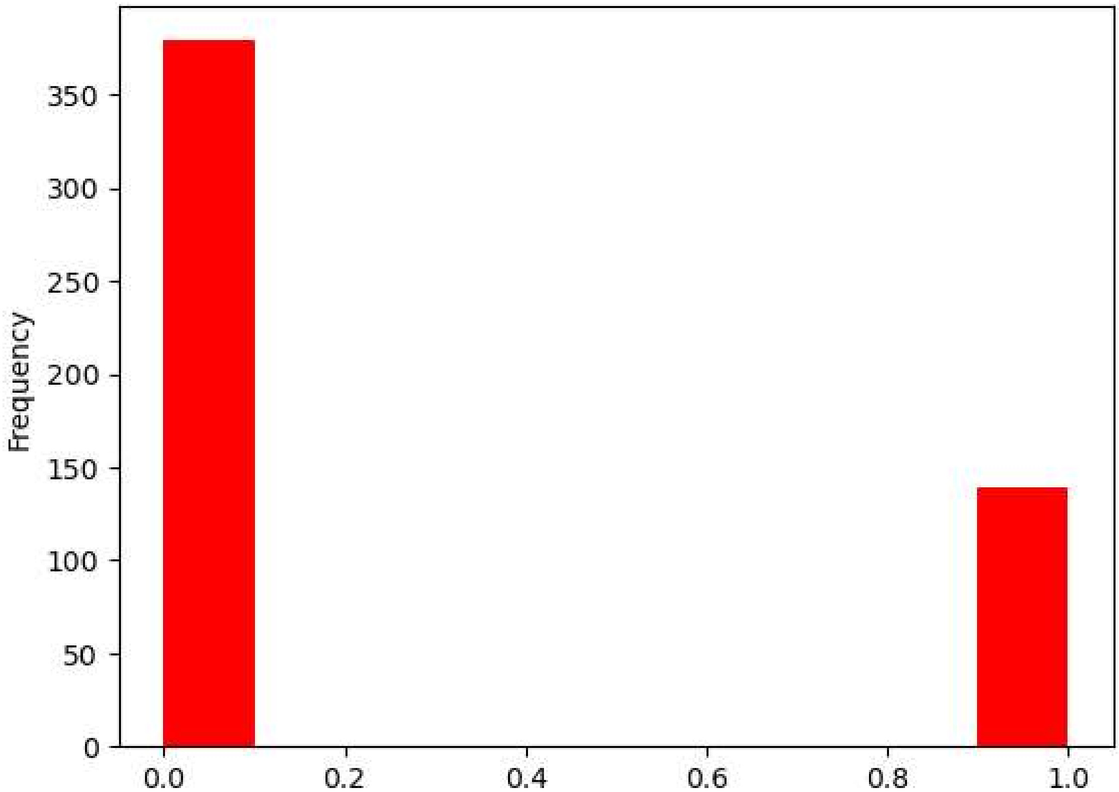
In [14]:
```
data['credit'].plot(kind='hist',color='orange')
```

Out[14]: <Axes: ylabel='Frequency'>



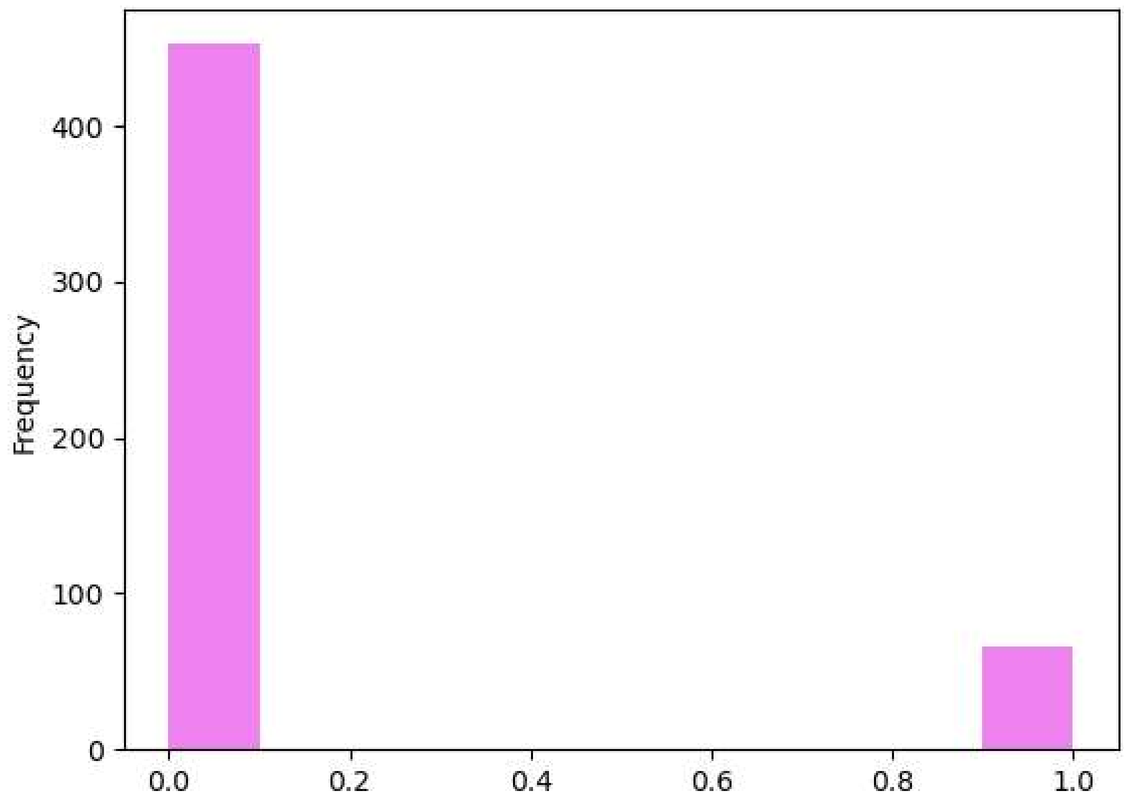
In [15]:
```
data['Gender'].plot(kind='hist',color='red')
```

Out[15]: <Axes: ylabel='Frequency'>



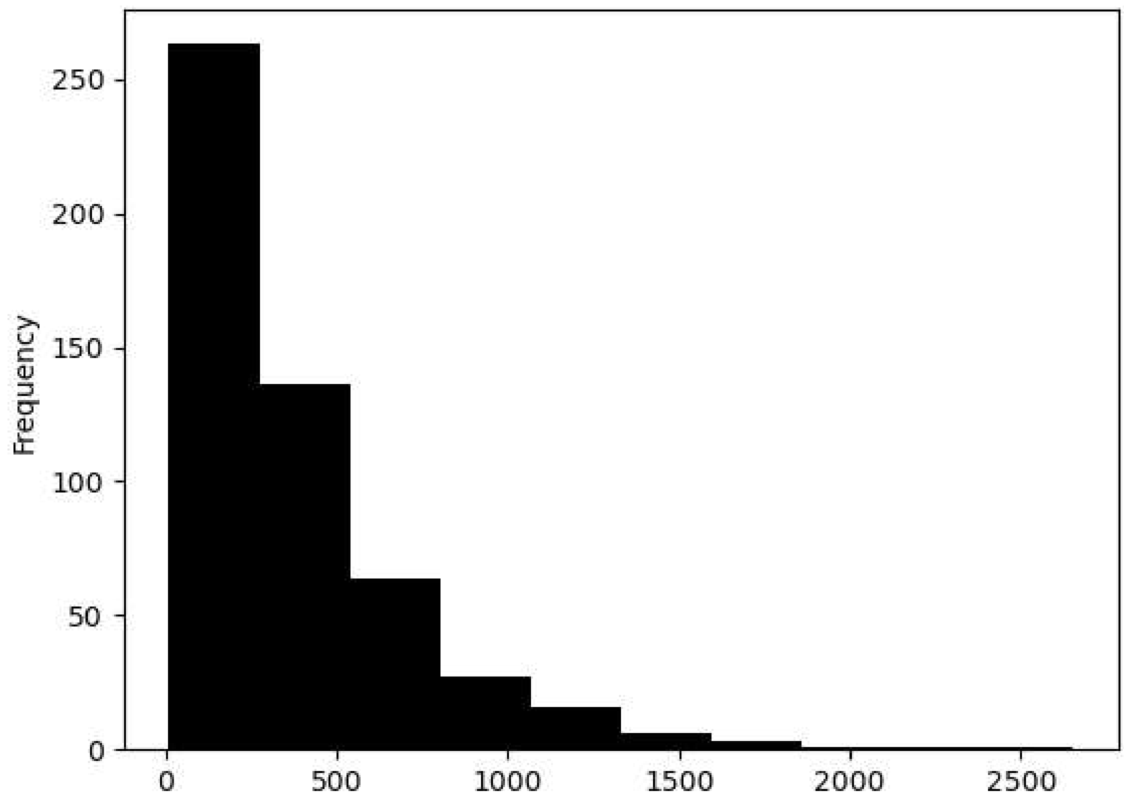
In [16]:
```
data['previous'].plot(kind='hist',color='violet')
```

Out[16]: <Axes: ylabel='Frequency'>



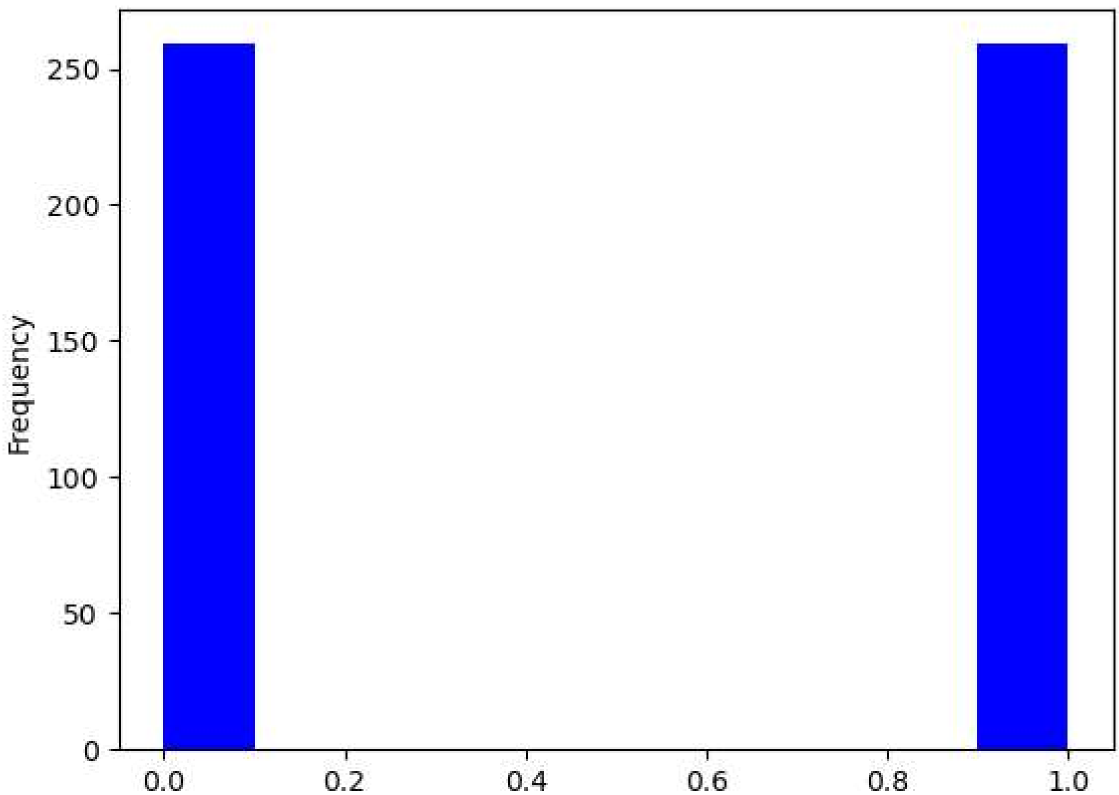
In [17]:
```
data['duration'].plot(kind='hist',color='black')
```

Out[17]: <Axes: ylabel='Frequency'>

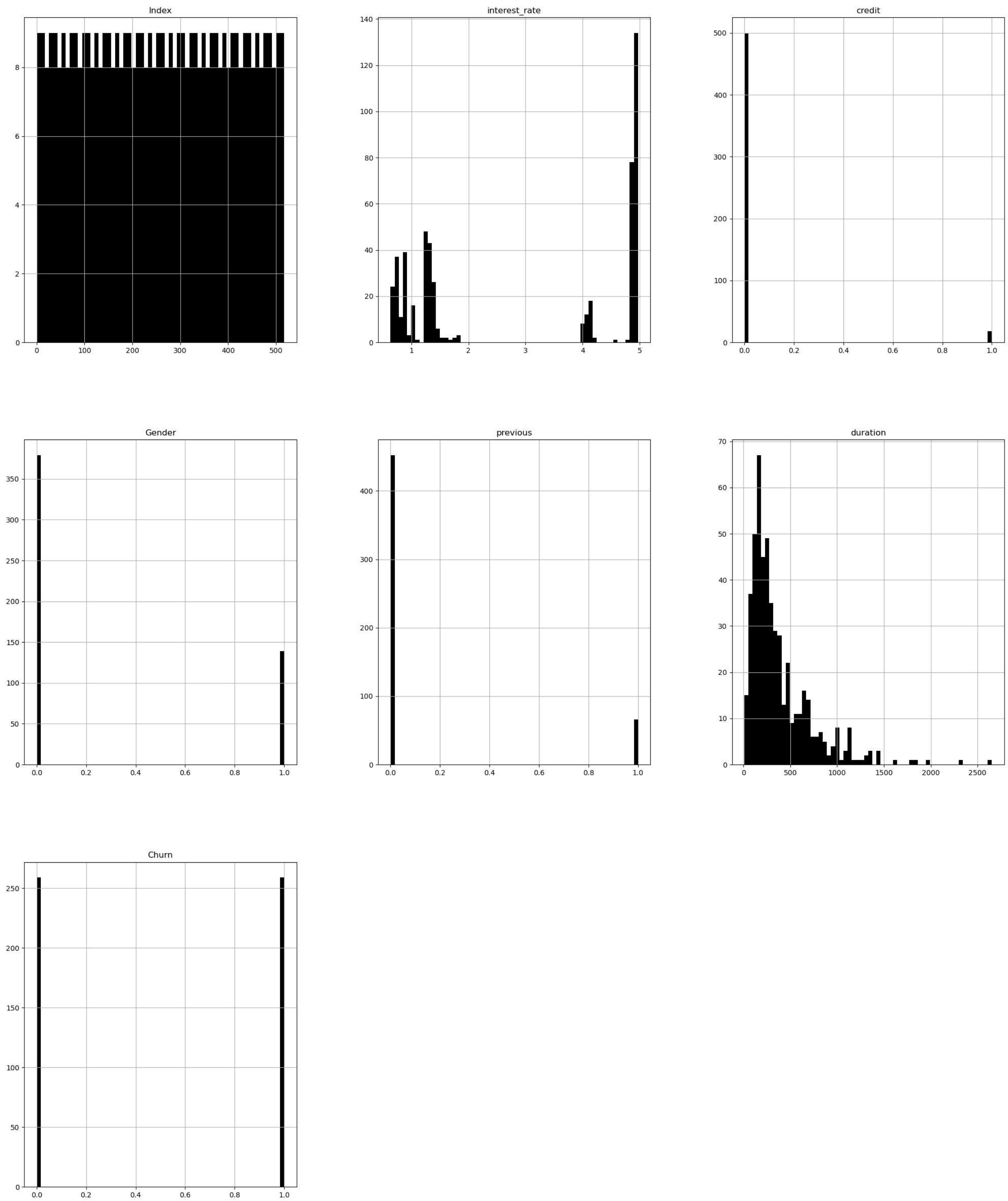


In [18]:
```
data['Churn'].plot(kind='hist',color='blue')
```

Out[18]:
```
<Axes: ylabel='Frequency'>
```



In [19]:
```
data.hist(bins=60,figsize=(25,30),color='black')
plt.show()
```



In [ ]:

In [20]:
```
for i in data.columns:
    print("*********************************************************",i,
          "*******************************************************")
    print()
    print(set(data[i].tolist()))
    print()
```

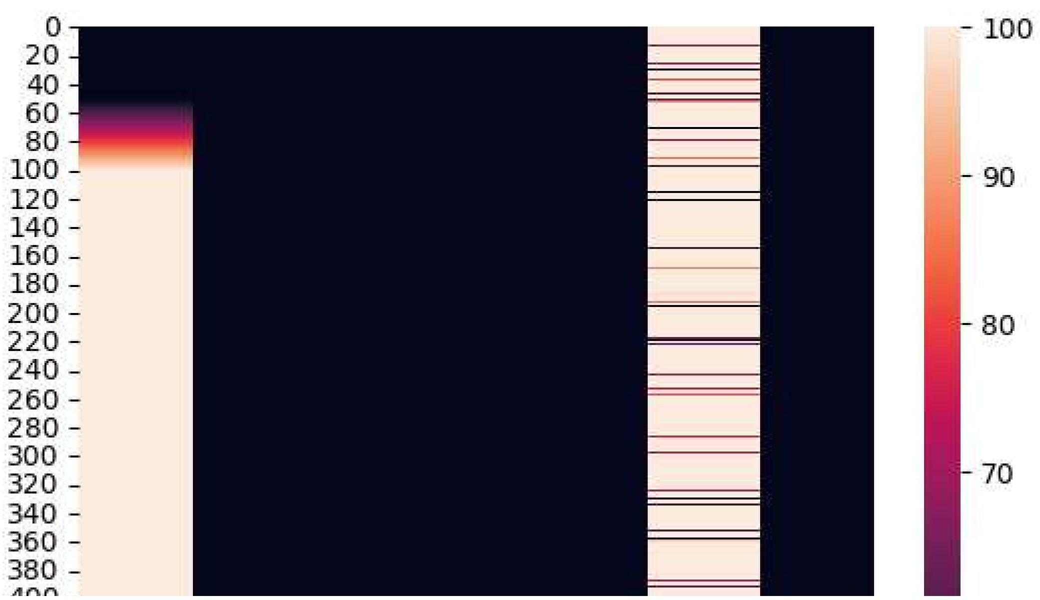
```
******************************************************* Index ************
*****************************************

{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 2
1, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 4
0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 5
9, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 7
8, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 9
7, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 1
13, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 12
8, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143,
144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 15
9, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174,
175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 19
0, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205,
206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 22
1, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236,
237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 25
2, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267,
268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 28
3, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298,
299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 31
4, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329,
330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 34
5, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360,
361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 37
6, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391,
392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 40
7, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422,
423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 43
8, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453,
454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 46
9, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484,
485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 50
0, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515,
516, 517}

******************************************************* interest_rate *****
**************************************************

{0.899, 1.334, 1.365, 1.244, 4.858, 4.12, 4.856, 4.962, 4.865, 4.965, 4.864,
4.86, 1.811, 4.964, 1.0, 1.25, 0.904, 0.683, 4.866, 0.898, 0.959, 0.677, 1.29
9, 1.799, 1.049, 0.982, 0.697, 0.851, 0.79, 1.04, 1.726, 1.354, 1.479, 1.415,
0.72, 4.592, 1.406, 4.857, 1.281, 4.97, 4.921, 4.153, 4.967, 1.531, 0.843, 0.
714, 1.028, 1.4, 0.827, 1.327, 1.266, 1.016, 0.635, 0.885, 4.961, 1.26, 4.95
8, 1.757, 4.955, 0.879, 0.748, 0.873, 1.687, 1.498, 0.742, 0.896, 0.771, 0.64
6, 0.861, 4.794, 1.05, 1.614, 0.73, 1.483, 1.291, 1.041, 0.849, 1.602, 1.41,
0.878, 0.767, 0.715, 0.773, 0.854, 0.668, 0.652, 0.739, 0.723, 0.884, 0.682,
1.029, 4.968, 4.855, 4.959, 0.706, 0.645, 1.27, 1.392, 1.52, 0.735, 0.639, 0.
889, 1.453, 1.264, 0.883, 1.072, 0.877, 4.191, 1.313, 1.252, 0.717, 0.81, 0.
9, 0.74, 0.644, 0.859, 1.423, 1.048, 0.699, 0.728, 4.859, 1.039, 0.882, 0.72
2, 0.655, 0.716, 1.405, 1.344, 1.03, 0.87, 0.835, 0.649, 0.704, 4.966, 4.076,
4.963, 4.957, 4.021, 1.268, 4.96, 0.797, 0.721, 1.262, 1.259, 0.977, 0.881,
0.846}

******************************************************* credit ***********
******************************************
```

```
{0, 1}

******************************************************** Gender ************
*******************************************

{0, 1}

******************************************************** previous **********
*********************************************

{0, 1}

******************************************************** duration **********
*********************************************

{9, 10, 11, 16, 25, 35, 39, 40, 42, 49, 51, 56, 57, 58, 59, 63, 64, 68, 69, 7
0, 72, 73, 74, 76, 80, 81, 82, 83, 85, 86, 87, 88, 95, 96, 97, 98, 99, 101, 1
02, 104, 105, 107, 109, 111, 113, 114, 115, 116, 117, 119, 120, 123, 124, 12
5, 126, 127, 128, 129, 130, 132, 133, 135, 136, 141, 142, 143, 144, 145, 146,
147, 148, 149, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 163, 16
4, 166, 167, 169, 170, 171, 172, 173, 174, 175, 178, 179, 180, 181, 183, 184,
185, 187, 188, 191, 192, 193, 194, 195, 196, 198, 199, 201, 202, 203, 204, 20
5, 206, 207, 208, 209, 211, 212, 213, 216, 217, 218, 219, 222, 223, 224, 225,
227, 228, 229, 230, 232, 233, 238, 239, 241, 243, 244, 247, 248, 249, 250, 25
1, 252, 2301, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 26
6, 267, 268, 269, 270, 272, 273, 274, 275, 276, 279, 280, 281, 285, 286, 288,
289, 290, 292, 293, 294, 295, 298, 299, 300, 301, 308, 309, 310, 311, 314, 31
6, 317, 320, 322, 323, 326, 328, 330, 331, 334, 336, 337, 338, 340, 341, 342,
344, 345, 346, 353, 355, 357, 360, 361, 362, 363, 364, 365, 367, 374, 375, 37
9, 383, 384, 387, 388, 389, 391, 393, 394, 395, 396, 398, 401, 403, 404, 405,
406, 408, 412, 417, 418, 423, 424, 436, 446, 449, 450, 454, 455, 456, 458, 45
9, 460, 461, 463, 470, 473, 474, 475, 476, 477, 479, 482, 483, 491, 496, 498,
519, 523, 525, 530, 531, 537, 539, 544, 545, 546, 549, 553, 565, 568, 569, 57
1, 578, 582, 585, 587, 592, 600, 604, 2653, 608, 609, 617, 619, 621, 626, 62
8, 629, 632, 633, 634, 640, 645, 650, 653, 655, 662, 663, 673, 683, 686, 688,
689, 690, 693, 697, 698, 706, 711, 712, 716, 728, 738, 739, 742, 747, 760, 76
2, 767, 768, 771, 796, 805, 806, 809, 815, 829, 840, 846, 854, 873, 886, 895,
898, 941, 951, 955, 958, 980, 984, 988, 1009, 1011, 1013, 1019, 1062, 1068, 1
087, 1091, 1127, 1130, 1135, 1138, 1143, 1148, 1150, 1152, 1176, 1234, 1276,
1311, 1319, 1340, 1348, 1357, 1422, 1424, 1447, 1602, 1806, 1855, 1980}

******************************************************** Churn *************
******************************************

{0, 1}
```

In [21]:
```
# Finding correlation
plt.figure(figsize=(20,15))
corr = data.corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.show()
```

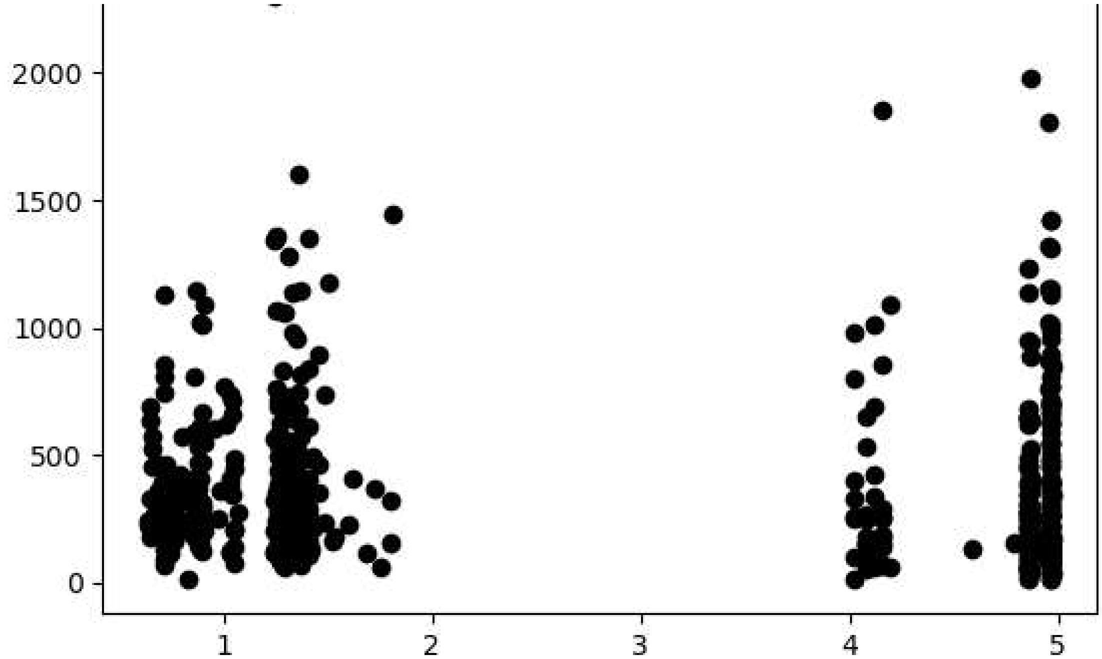
In [22]:
```
sns.heatmap(data, vmin=50, vmax=100)
```

Out[22]: <Axes: >



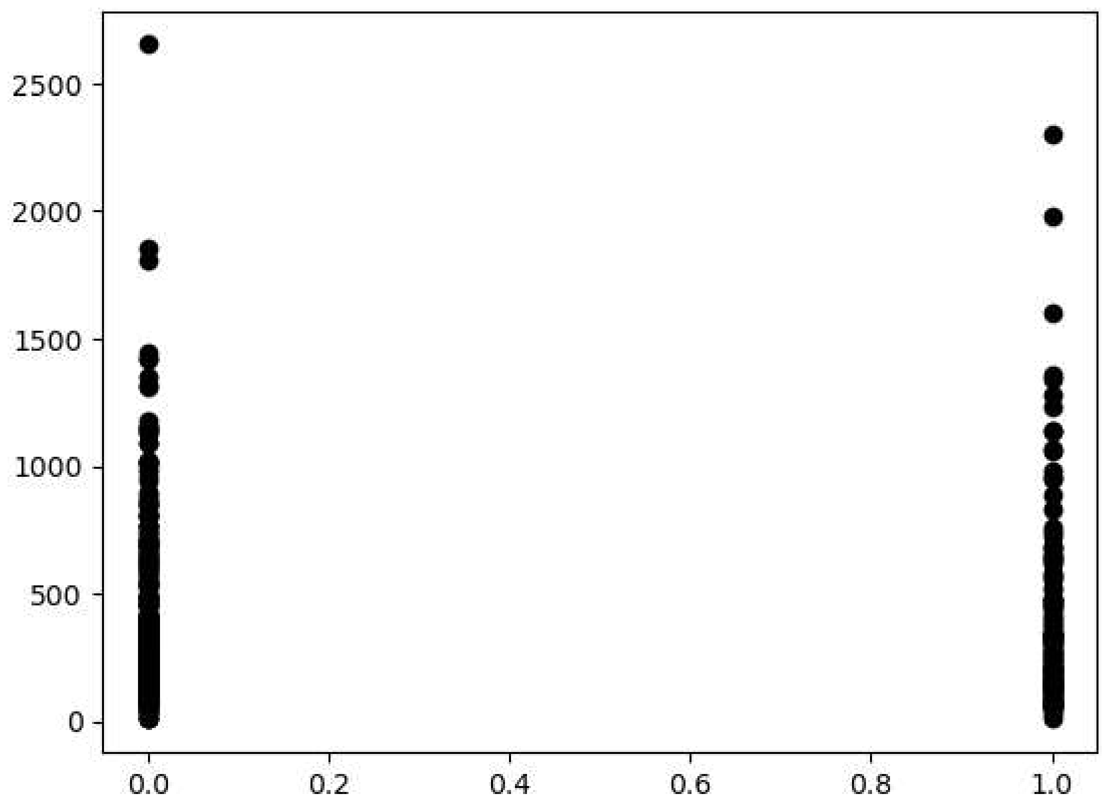
In [ ]:

In [24]:
```
plt.scatter(data['interest rate'],data['duration'],color='black')
```



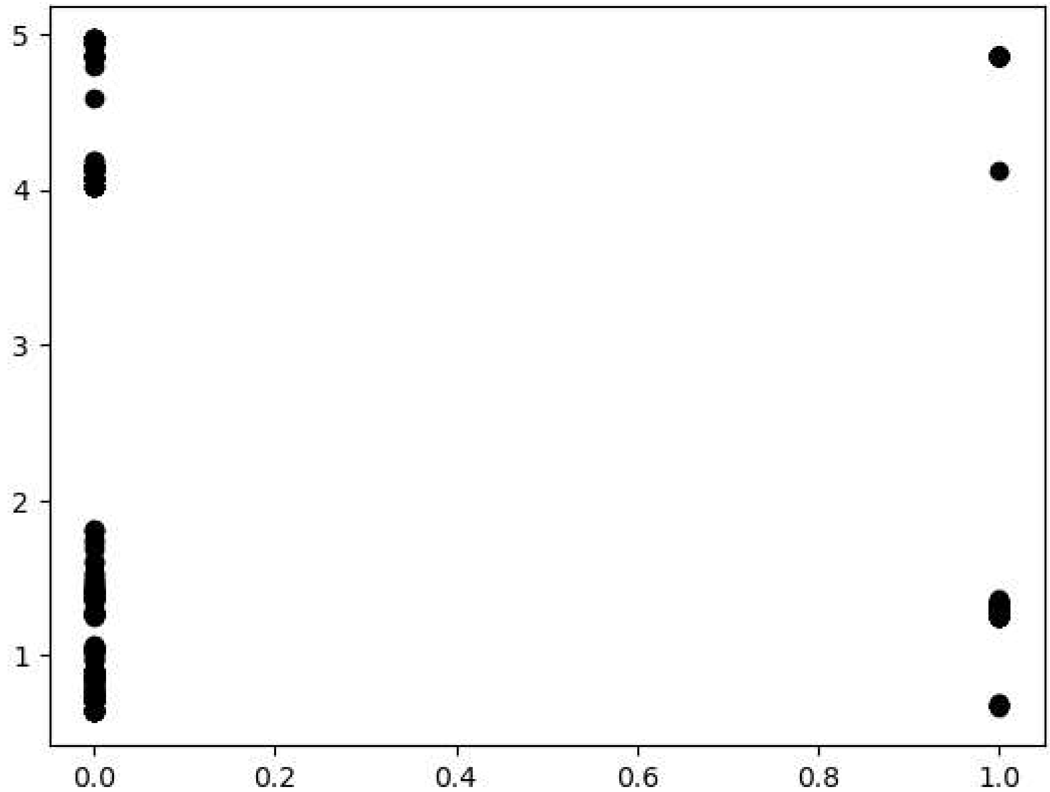
In [25]:
```
plt.scatter(data['Gender'],data['duration'],color='black')
```

Out[25]: <matplotlib.collections.PathCollection at 0x25fba4baa40>



In [26]:
```
plt.scatter(data['Gender'],data['interest rate'],color='black')
```

Out[26]: <matplotlib.collections.PathCollection at 0x25fba67de40>



In [27]:
```
x=data.drop(['Index','credit','Gender','previous','Churn'],axis=1).values
x
```

Out[27]:
```
array([[  1.334, 117.   ],
       [  0.767, 274.   ],
       [  4.858, 167.   ],
       ...,
       [  0.879, 290.   ],
       [  0.877, 473.   ],
       [  4.965, 142.   ]])
```

scale the data

In [28]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc_x = sc.fit_transform(x)
sc_x
```

Out[28]:
```
array([[-0.83643824, -0.77850438],
       [-1.13856587, -0.32116532],
       [ 1.04133564, -0.632855  ],
       ...,
       [-1.07888634, -0.27455752],
       [-1.07995205,  0.25851922],
       [ 1.0983509 , -0.70567969]])
```

Use K Means Cluster

In [29]:
```
from sklearn.cluster import KMeans
```

In [30]:
```
k_means = KMeans(n_clusters=3, random_state=1)
k_means.fit(sc_x)
```

Out[30]:
```
KMeans(n_clusters=3, random_state=1)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [31]:
```
k_means.labels_
```

Out[31]:
```
array([1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 2, 0, 2, 1, 0, 1, 0, 0,
       1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1,
       0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 2, 1, 1,
       1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 2, 0, 2, 0, 1, 1, 0, 1, 0, 1, 1, 0,
       1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 2, 1, 2, 1, 1, 1, 1, 0, 1,
       1, 2, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 2, 1, 2, 0, 2, 1,
       0, 1, 0, 2, 1, 2, 1, 1, 2, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 2,
       1, 1, 1, 1, 0, 1, 0, 2, 0, 0, 0, 0, 2, 1, 1, 1, 1, 1, 1, 0, 1, 0,
       1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 2, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 2, 2, 0, 0, 0, 2, 0, 2, 1, 1, 1, 1, 0, 1,
       0, 1, 2, 2, 1, 0, 2, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 2, 0,
       1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 2, 1, 1, 1, 0, 1, 2, 1, 2, 2, 0, 2,
       0, 2, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 2,
       1, 1, 0, 1, 0, 1, 2, 1, 2, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 2, 1,
       2, 2, 2, 1, 0, 1, 1, 1, 1, 1, 1, 0, 2, 0, 0, 1, 1, 2, 1, 0, 1, 1,
       0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 2, 0, 1, 0, 0, 2, 2, 1, 1,
       1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 2, 1,
       1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 2, 1, 2, 0, 1, 1, 0, 0, 1, 1, 0,
       0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 2, 0, 1, 1, 0, 0, 0, 0, 1, 0, 2,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 2, 0, 1, 1, 0, 1,
       1, 0, 0, 0, 0, 2, 1, 1, 1, 0, 0, 2, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0,
       2, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0,
       1, 0, 2, 1, 1, 0, 1, 1, 1, 1, 1, 0])
```

Use Elbow method

In [32]:
```
wcss = []
k = list(range(1, 15))
for i in k:
    kmeans = KMeans(n_clusters = i)
    kmeans.fit(sc_x)
    wcss.append(kmeans.inertia_)
    print(f'For n_clusters = {i}, the intertia is {kmeans.inertia_}' )
```

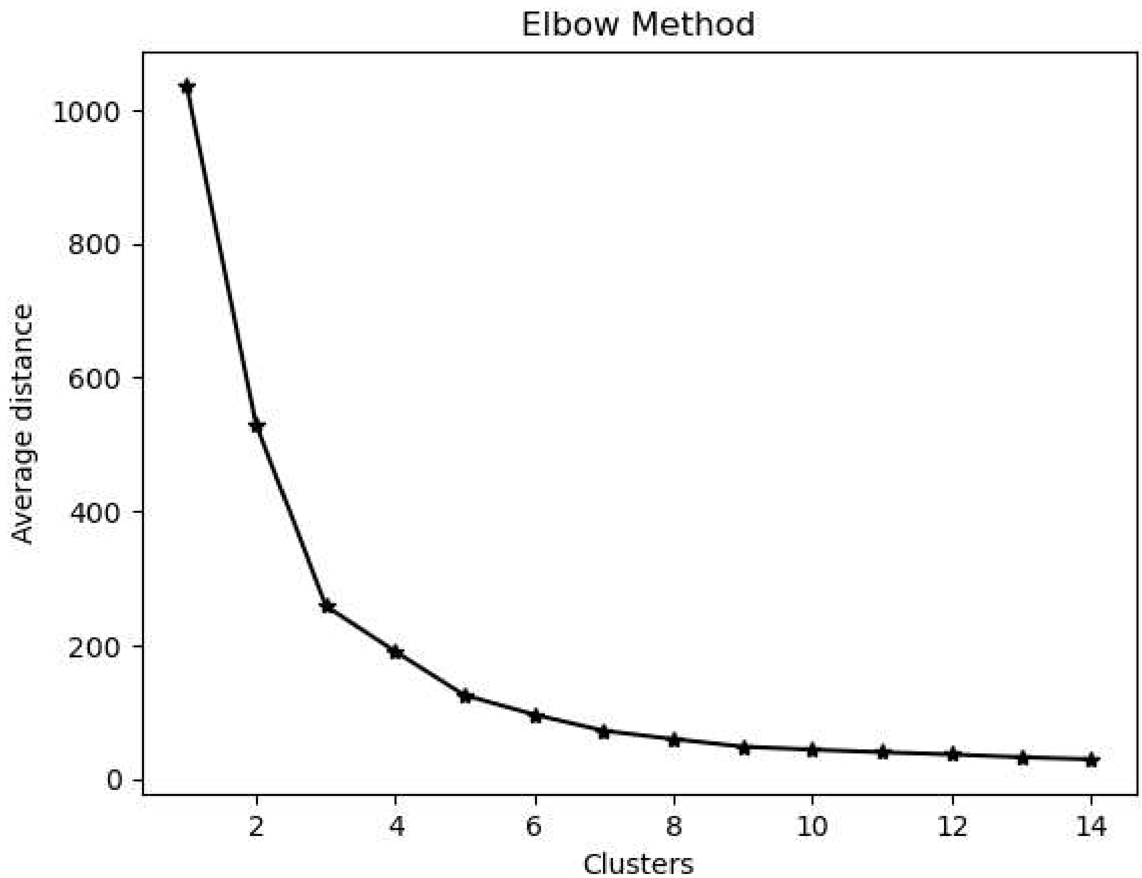
```
For n_clusters = 1, the intertia is 1036.0
For n_clusters = 2, the intertia is 530.2844942612068
For n_clusters = 3, the intertia is 258.80051758779
For n_clusters = 4, the intertia is 190.72628746155505
For n_clusters = 5, the intertia is 125.10920333382882
For n_clusters = 6, the intertia is 96.12221236262565
For n_clusters = 7, the intertia is 72.00247982978159
For n_clusters = 8, the intertia is 59.550255519346585
For n_clusters = 9, the intertia is 48.214497939909464
For n_clusters = 10, the intertia is 43.76816097195865
For n_clusters = 11, the intertia is 39.81177225457489
For n_clusters = 12, the intertia is 36.60731705522372
For n_clusters = 13, the intertia is 32.39815522108192
For n_clusters = 14, the intertia is 29.306876754440772
```

In [33]:
```
wcss
```

Out[33]:
```
[1036.0,
 530.2844942612068,
 258.80051758779,
 190.72628746155505,
 125.10920333382882,
 96.12221236262565,
 72.00247982978159,
 59.550255519346585,
 48.214497939909464,
 43.76816097195865,
 39.81177225457489,
 36.60731705522372,
 32.39815522108192,
 29.306876754440772]
```

In [34]:
```
plt.plot(k, wcss, color= 'black', marker='*')
plt.xlabel('Clusters')
plt.ylabel('Average distance')
plt.title('Elbow Method')
```

Out[34]:
```
Text(0.5, 1.0, 'Elbow Method')
```



Basis the elbow method, we can say K = 3

In [35]:
```
k_means = KMeans(n_clusters=3, random_state=1)
k_means.fit(sc_x)
labels = k_means.labels_
labels
```

Out[35]:
```
array([1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 2, 0, 2, 1, 0, 1, 0, 0,
       1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1,
       0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 2, 1, 1,
       1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 2, 0, 2, 0, 1, 1, 0, 1, 0, 1, 1, 0,
       1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 2, 1, 2, 1, 1, 1, 1, 0, 1,
       1, 2, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 2, 1, 2, 0, 2, 1,
       0, 1, 0, 2, 1, 2, 1, 1, 2, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 2,
       1, 1, 1, 1, 0, 1, 0, 2, 0, 0, 0, 0, 2, 1, 1, 1, 1, 1, 1, 0, 1, 0,
       1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 2, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 2, 2, 0, 0, 0, 2, 0, 2, 1, 1, 1, 1, 0, 1,
       0, 1, 2, 2, 1, 0, 2, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 2, 0,
       1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 2, 1, 1, 1, 0, 1, 2, 1, 2, 2, 0, 2,
       0, 2, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 2,
       1, 1, 0, 1, 0, 1, 2, 1, 2, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 2, 1,
       2, 2, 2, 1, 0, 1, 1, 1, 1, 1, 1, 0, 2, 0, 0, 1, 1, 2, 1, 0, 1, 1,
       0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 2, 0, 1, 0, 0, 2, 2, 1, 1,
       1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 2, 1,
       1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 2, 1, 2, 0, 1, 1, 0, 0, 1, 1, 0,
       0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 2, 0, 1, 1, 0, 0, 0, 0, 1, 0, 2,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 2, 0, 1, 1, 0, 1,
       1, 0, 0, 0, 0, 2, 1, 1, 1, 0, 0, 2, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0,
       2, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0,
       1, 0, 2, 1, 1, 0, 1, 1, 1, 1, 1, 0])
```

In [36]:
```
y_kmeans = k_means.fit_predict(x)
y_kmeans
```

Out[36]:
```
array([0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 2, 0, 2, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 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, 1, 1, 1, 0, 2, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 2, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 2, 0, 1, 1, 2, 1,
       0, 0, 1, 2, 1, 2, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 2,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 2, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 2, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 2, 2, 1, 0, 0, 1, 1, 2, 0, 0, 0, 0, 0, 1,
       0, 0, 2, 2, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 2, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 2, 0, 1, 0, 0, 0, 1, 0, 1, 2, 0, 1,
       0, 2, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 2,
       1, 0, 0, 0, 0, 0, 1, 0, 2, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0,
       2, 1, 2, 0, 0, 0, 1, 0, 1, 1, 1, 0, 2, 0, 0, 0, 1, 2, 0, 0, 0, 1,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 0, 2, 2, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 2, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 2, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 2, 0, 0, 1, 1, 0, 2, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       2, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0])
```

In [37]:
```
# Evaluation for all 3 Cluster : find the silhouett score
from sklearn.metrics import silhouette_samples, silhouette_score
```

In [38]:
```
silhouette_score(sc_x, labels, random_state=1)
```

Out[38]:
```
0.6415946769589214
```

In [39]:
```
data.head()
```

Out[39]:

	Index	interest_rate	credit	Gender	previous	duration	Churn
0	0	1.334	0	1	0	117	0
1	1	0.767	0	0	1	274	1
2	2	4.858	0	1	0	167	0
3	3	4.120	0	0	0	686	1
4	4	4.856	0	1	0	159	0

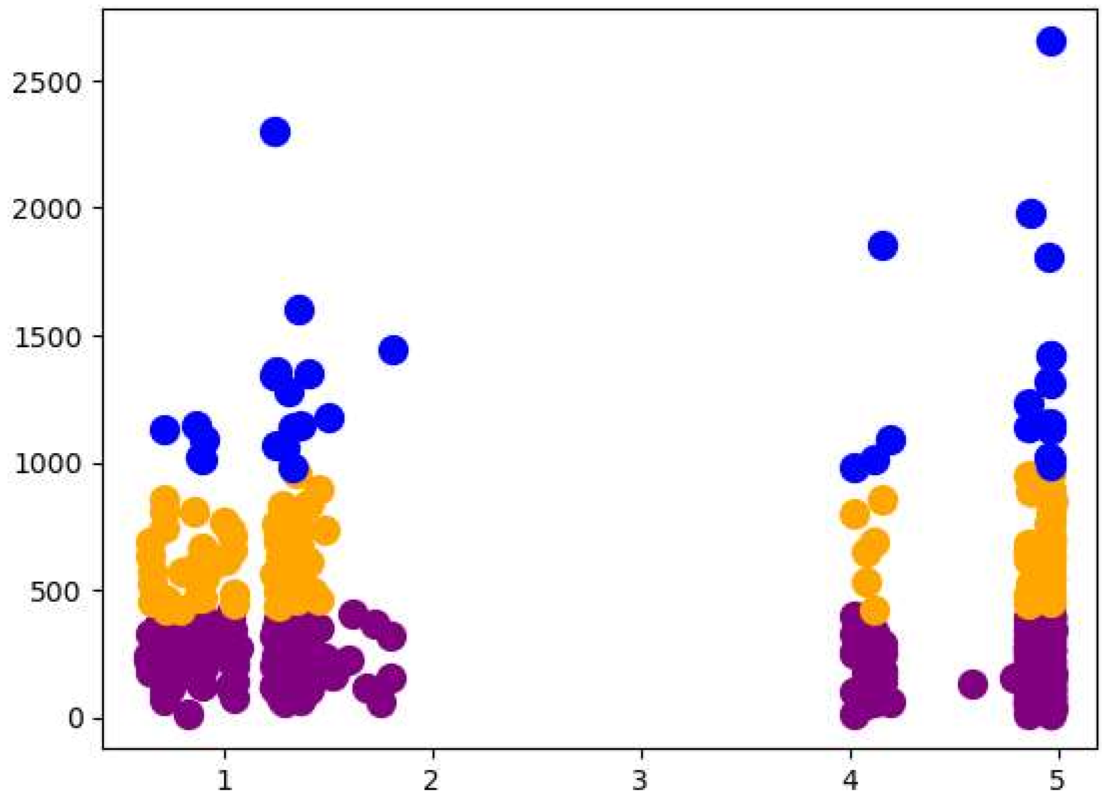
In [40]:
```
data['Cluster_Kmeans_3'] = labels
data.head()
```

Out[40]:

	Index	interest_rate	credit	Gender	previous	duration	Churn	Cluster_Kmeans_3
0	0	1.334	0	1	0	117	0	1
1	1	0.767	0	0	1	274	1	1
2	2	4.858	0	1	0	167	0	0
3	3	4.120	0	0	0	686	1	0
4	4	4.856	0	1	0	159	0	0

Visualization

In [41]:
```
plt.scatter(x[y_kmeans==0,0], x[y_kmeans==0,1], s=100, c='purple',label='Clust
plt.scatter(x[y_kmeans==1,0], x[y_kmeans==1,1], s=100, c='orange',label='Clust
plt.scatter(x[y_kmeans==2,0], x[y_kmeans==2,1], s=100, c='blue',label='Cluster
plt.show()
```



In [ ]:

K = 4

In [42]:
```
k_means = KMeans(n_clusters=4, random_state=1)
k_means.fit(sc_x)
labels = k_means.labels_
labels
```

Out[42]:
```
array([0, 0, 1, 3, 1, 0, 1, 3, 3, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 3, 0, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 3, 0, 1, 1, 2, 1, 2, 0, 1, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 3, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 3, 0, 0,
       0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 3, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 2, 0, 2, 0, 0, 0, 0, 1, 0,
       0, 3, 0, 1, 1, 0, 1, 1, 1, 1, 3, 0, 3, 0, 0, 1, 2, 0, 3, 3, 3, 0,
       1, 0, 3, 2, 0, 2, 0, 0, 3, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 3,
       0, 0, 0, 0, 1, 0, 1, 3, 1, 1, 1, 1, 2, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 1, 3, 0, 0, 1, 1, 0, 3, 1, 2, 0, 1, 0, 0, 1, 3, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 3, 0, 2, 3, 1, 1, 1, 3, 3, 2, 0, 0, 0, 0, 1, 0,
       1, 0, 2, 2, 0, 1, 3, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 2, 1,
       0, 1, 0, 0, 3, 1, 0, 0, 3, 1, 2, 0, 0, 0, 1, 0, 3, 0, 3, 2, 1, 3,
       1, 3, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 3,
       0, 0, 1, 0, 1, 0, 3, 0, 3, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 2, 0,
       2, 3, 2, 0, 1, 0, 0, 0, 0, 0, 0, 1, 2, 1, 1, 0, 0, 2, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 3, 1, 0, 1, 1, 2, 3, 0, 0,
       0, 1, 1, 1, 0, 0, 0, 0, 1, 3, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 3, 0,
       0, 1, 1, 3, 0, 0, 1, 0, 1, 1, 1, 2, 0, 3, 1, 0, 0, 1, 1, 0, 0, 1,
       3, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 3, 3, 0, 0, 1, 1, 1, 1, 0, 1, 2,
       1, 1, 1, 1, 0, 1, 0, 1, 3, 1, 3, 0, 0, 0, 1, 0, 2, 3, 0, 0, 1, 0,
       0, 1, 1, 1, 1, 2, 0, 0, 0, 1, 1, 2, 0, 0, 0, 0, 1, 3, 1, 0, 1, 1,
       2, 0, 0, 3, 0, 1, 0, 3, 3, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 3,
       0, 1, 3, 0, 0, 1, 0, 0, 0, 0, 0, 1])
```

In [43]:
```
y_kmeans = k_means.fit_predict(x)
y_kmeans
```

Out[43]:
```
array([0, 0, 0, 3, 0, 0, 0, 3, 3, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 3, 3,
       0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 3, 0, 0, 0, 1, 0, 1, 3, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 3, 0, 0, 0,
       0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 3, 3, 3, 0, 1, 0, 1, 0, 3, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 3, 0, 0, 3, 3, 0, 0, 0, 0, 1, 0, 1, 0, 3, 0, 0, 0, 0,
       0, 3, 0, 3, 3, 0, 0, 0, 0, 0, 3, 0, 3, 3, 0, 0, 2, 0, 1, 3, 1, 3,
       0, 0, 3, 1, 3, 1, 0, 3, 1, 0, 0, 0, 0, 0, 0, 3, 3, 0, 0, 0, 0, 1,
       0, 0, 3, 0, 0, 0, 0, 1, 0, 0, 0, 0, 2, 0, 0, 3, 0, 3, 0, 0, 3, 0,
       3, 0, 0, 3, 0, 1, 0, 0, 0, 3, 0, 1, 0, 0, 1, 0, 0, 3, 3, 0, 0, 0,
       3, 3, 0, 0, 3, 0, 3, 0, 1, 1, 3, 0, 0, 1, 3, 1, 0, 0, 0, 3, 0, 3,
       0, 0, 1, 1, 3, 0, 1, 3, 0, 3, 0, 0, 0, 3, 0, 0, 0, 3, 3, 3, 1, 0,
       0, 0, 0, 0, 3, 3, 3, 0, 3, 0, 1, 0, 3, 0, 0, 0, 3, 0, 1, 2, 0, 1,
       0, 1, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 1,
       3, 0, 0, 0, 0, 0, 1, 0, 1, 0, 3, 3, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 1, 1, 0, 0, 0, 3, 0, 3, 3, 3, 0, 1, 0, 0, 0, 3, 2, 0, 0, 3, 3,
       0, 3, 3, 0, 3, 0, 0, 0, 0, 0, 0, 0, 3, 1, 0, 0, 3, 0, 2, 1, 0, 3,
       3, 0, 0, 0, 1, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 3, 0, 0, 3, 0, 1, 0,
       0, 0, 3, 3, 0, 0, 3, 3, 0, 0, 0, 1, 0, 1, 3, 3, 3, 3, 0, 0, 3, 0,
```

In [44]:
```
from sklearn.metrics import silhouette_samples, silhouette_score
```

In [45]:
```
silhouette_score(sc_x, labels, random_state=1)
```

Out[45]: 0.6498972328936907

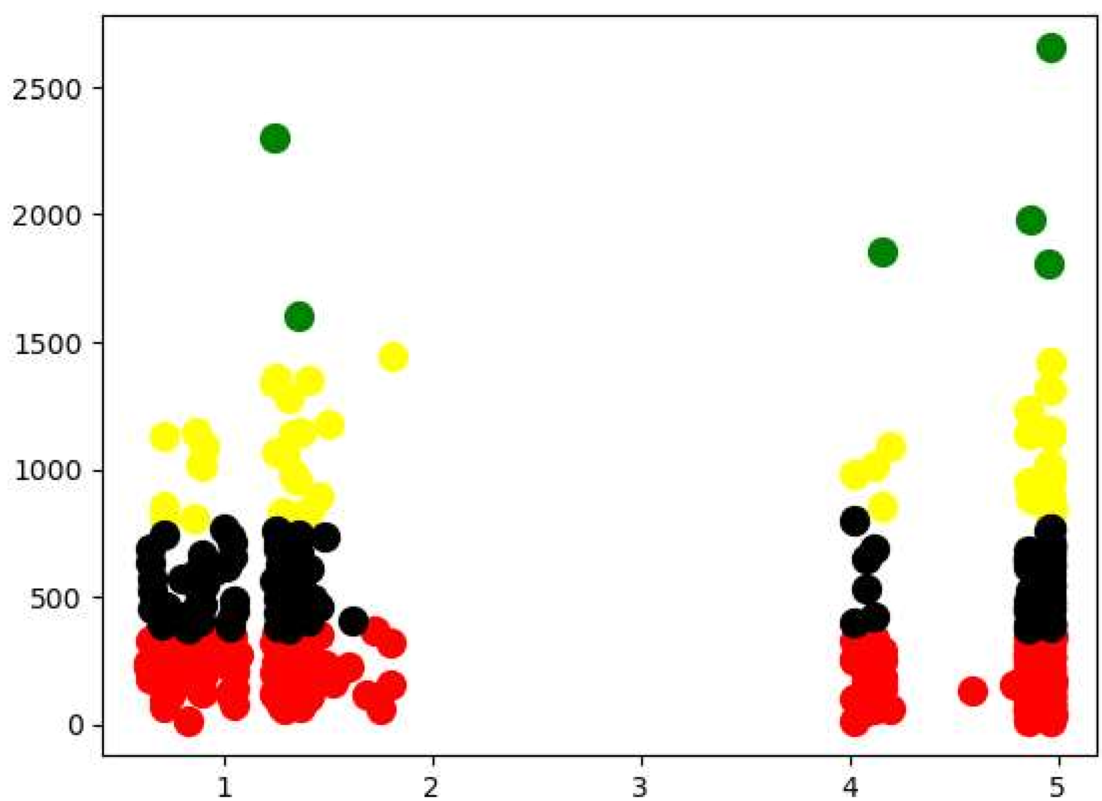
In [46]:
```
data.head()
```

Out[46]:

	Index	interest_rate	credit	Gender	previous	duration	Churn	Cluster_Kmeans_3
0	0	1.334	0	1	0	117	0	1
1	1	0.767	0	0	1	274	1	1
2	2	4.858	0	1	0	167	0	0
3	3	4.120	0	0	0	686	1	0
4	4	4.856	0	1	0	159	0	0

Visualization

In [47]:
```
plt.scatter(x[y_kmeans==0,0], x[y_kmeans==0,1], s=100, c='red',label='Cluster1
plt.scatter(x[y_kmeans==1,0], x[y_kmeans==1,1], s=100, c='yellow',label='Clust
plt.scatter(x[y_kmeans==2,0], x[y_kmeans==2,1], s=100, c='green',label='Cluste
plt.scatter(x[y_kmeans==3,0], x[y_kmeans==3,1], s=100, c='black',label='Cluste
plt.show()
```



K = 5

In [48]:
```
k_means = KMeans(n_clusters=5, random_state=1)
k_means.fit(sc_x)
labels = k_means.labels_
labels
```

Out[48]:
```
array([0, 0, 1, 2, 1, 0, 1, 2, 2, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 2, 3, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 2, 0, 1, 1, 3, 1, 4, 0, 1, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 2, 1, 1, 1, 1, 0, 1, 0, 3, 0, 1, 0, 3, 0, 1, 0,
       1, 1, 3, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 2, 0, 0,
       0, 1, 1, 1, 0, 0, 3, 3, 3, 1, 2, 1, 3, 1, 3, 0, 1, 0, 1, 0, 0, 1,
       0, 0, 0, 1, 3, 1, 1, 3, 0, 1, 0, 1, 0, 4, 0, 3, 0, 0, 0, 0, 1, 0,
       0, 2, 0, 1, 1, 0, 1, 1, 1, 1, 2, 0, 2, 0, 0, 1, 4, 0, 2, 2, 2, 3,
       1, 0, 2, 3, 3, 3, 0, 0, 2, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 2,
       0, 0, 0, 0, 1, 0, 1, 2, 1, 1, 1, 1, 4, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 1, 2, 0, 3, 1, 1, 0, 2, 1, 3, 0, 1, 3, 0, 1, 2, 3, 0, 0, 1,
       0, 3, 1, 1, 3, 1, 2, 0, 4, 2, 1, 1, 1, 2, 2, 3, 0, 0, 0, 0, 1, 3,
       1, 0, 3, 4, 0, 1, 2, 3, 1, 3, 0, 1, 1, 0, 0, 0, 0, 1, 0, 3, 4, 1,
       0, 1, 0, 0, 2, 1, 3, 0, 2, 1, 4, 0, 3, 0, 1, 0, 2, 0, 2, 4, 1, 2,
       1, 2, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 3, 1, 0, 1, 1, 1, 0, 1, 2,
       3, 0, 1, 0, 1, 0, 2, 0, 2, 0, 3, 1, 0, 1, 1, 1, 1, 0, 0, 0, 4, 0,
       4, 2, 4, 0, 1, 0, 3, 0, 3, 0, 0, 1, 3, 1, 1, 0, 3, 4, 0, 1, 0, 3,
       1, 0, 3, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 2, 1, 0, 1, 1, 4, 2, 0, 0,
       0, 1, 1, 1, 3, 0, 0, 0, 1, 2, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 2, 0,
       0, 1, 1, 2, 0, 0, 1, 3, 1, 1, 1, 3, 0, 2, 1, 3, 3, 1, 1, 0, 0, 1,
       2, 1, 3, 0, 1, 1, 0, 1, 0, 0, 0, 2, 2, 0, 0, 1, 1, 1, 1, 0, 1, 3,
       1, 1, 1, 1, 3, 1, 0, 1, 2, 1, 2, 3, 0, 3, 1, 0, 3, 2, 0, 0, 1, 3,
       0, 1, 1, 1, 1, 4, 0, 0, 0, 1, 1, 3, 0, 0, 0, 0, 1, 2, 1, 0, 1, 1,
       3, 3, 3, 2, 3, 1, 0, 2, 2, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 2,
       0, 1, 2, 0, 0, 1, 0, 0, 3, 0, 0, 1])
```

In [49]:
```
y_kmeans = k_means.fit_predict(x)
y_kmeans
```

Out[49]:
```
array([4, 1, 4, 3, 4, 4, 1, 3, 3, 1, 1, 1, 1, 4, 1, 1, 4, 4, 1, 3, 3, 1,
       1, 1, 1, 4, 1, 1, 1, 4, 4, 4, 3, 1, 4, 4, 0, 4, 0, 1, 4, 1, 4, 4,
       4, 4, 4, 4, 4, 1, 3, 4, 4, 4, 4, 1, 4, 4, 3, 1, 4, 4, 3, 1, 1, 1,
       4, 4, 3, 4, 4, 4, 1, 4, 1, 4, 1, 1, 4, 4, 4, 1, 4, 1, 1, 0, 4, 1,
       1, 4, 4, 4, 4, 4, 3, 3, 3, 4, 0, 4, 3, 4, 3, 1, 1, 1, 1, 4, 1, 1,
       1, 4, 4, 4, 3, 4, 4, 3, 1, 1, 4, 4, 1, 0, 1, 0, 4, 1, 4, 1, 4, 4,
       1, 3, 1, 1, 1, 1, 1, 4, 4, 4, 3, 1, 3, 1, 4, 4, 2, 4, 0, 3, 0, 3,
       4, 4, 3, 0, 3, 0, 1, 1, 3, 1, 4, 1, 4, 4, 4, 1, 1, 1, 1, 4, 4, 0,
       4, 4, 1, 4, 1, 4, 1, 3, 4, 4, 4, 4, 2, 4, 4, 1, 4, 1, 4, 4, 1, 4,
       1, 1, 4, 3, 1, 3, 4, 1, 1, 3, 1, 0, 4, 1, 3, 1, 4, 3, 3, 4, 4, 4,
       1, 3, 4, 4, 3, 4, 3, 4, 0, 0, 1, 4, 1, 3, 3, 0, 4, 4, 1, 1, 4, 3,
       4, 4, 0, 0, 1, 4, 3, 3, 4, 3, 4, 4, 4, 1, 4, 4, 1, 1, 1, 3, 0, 1,
       4, 4, 1, 4, 3, 1, 3, 1, 3, 4, 0, 1, 3, 4, 1, 4, 3, 4, 3, 2, 1, 0,
       4, 0, 1, 4, 4, 1, 1, 4, 1, 4, 4, 4, 4, 3, 1, 4, 4, 1, 4, 4, 4, 0,
       3, 4, 4, 4, 4, 1, 0, 4, 0, 1, 3, 1, 4, 4, 4, 4, 4, 4, 1, 4, 0, 4,
       0, 3, 0, 4, 4, 4, 3, 4, 3, 1, 1, 4, 0, 1, 4, 4, 3, 2, 4, 4, 1, 3,
       4, 1, 3, 4, 1, 4, 4, 4, 4, 4, 4, 4, 1, 0, 1, 1, 1, 4, 0, 0, 1, 1,
       1, 4, 4, 4, 3, 1, 1, 4, 4, 3, 4, 4, 1, 4, 4, 1, 4, 4, 1, 4, 0, 4,
       1, 4, 1, 3, 1, 4, 1, 3, 4, 4, 1, 0, 1, 3, 1, 3, 3, 1, 1, 1, 1, 4,
       3, 1, 3, 1, 1, 4, 1, 1, 1, 4, 1, 3, 3, 4, 1, 1, 1, 4, 4, 1, 1, 3,
       4, 4, 4, 4, 3, 1, 1, 1, 3, 4, 3, 3, 4, 3, 4, 4, 0, 3, 4, 4, 4, 3,
       1, 4, 4, 4, 4, 2, 1, 4, 1, 1, 1, 0, 1, 4, 4, 1, 4, 3, 4, 1, 1, 4,
       0, 3, 3, 3, 3, 1, 4, 3, 3, 1, 4, 4, 4, 4, 1, 4, 1, 4, 4, 4, 4, 1,
       1, 4, 3, 1, 1, 1, 4, 4, 3, 1, 1, 4])
```

In [ ]:

In [50]:
```
# Evaluation for all 5 Cluster : find the silhouett score
from sklearn.metrics import silhouette_samples, silhouette_score
```

In [51]:
```
silhouette_score(sc_x, labels, random_state=1)
```

Out[51]:
```
0.6307603699510385
```

In [ ]:

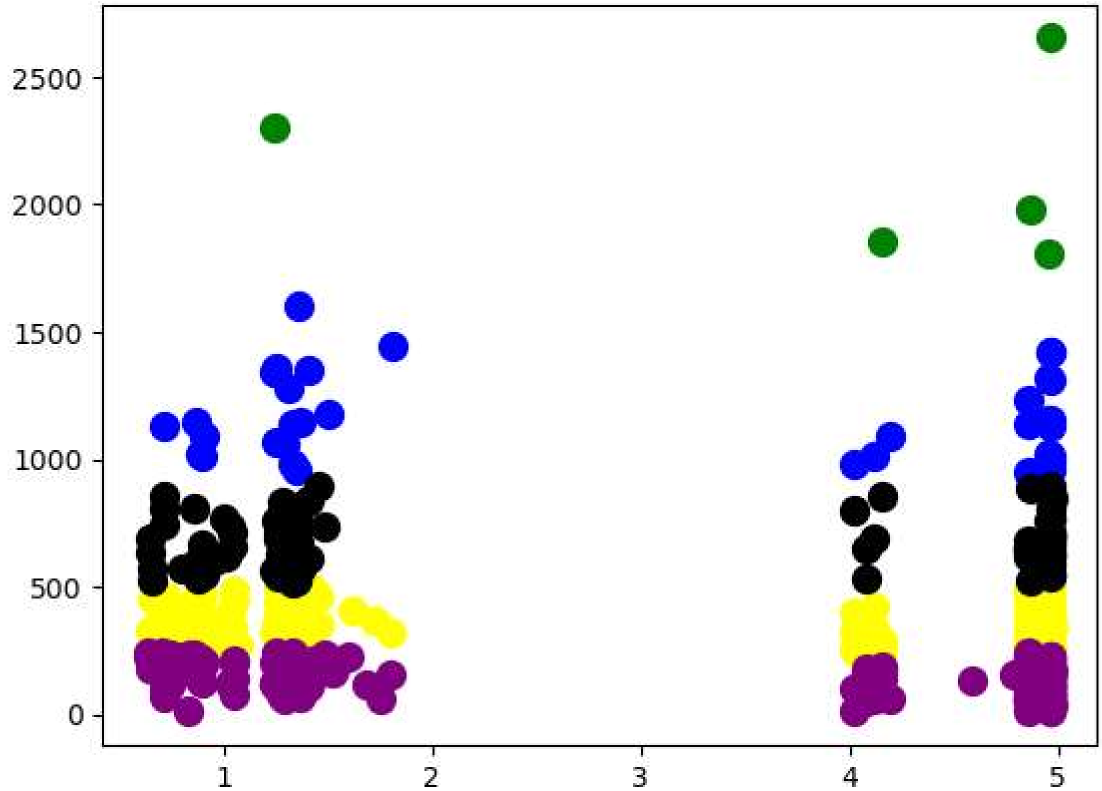
In [52]:
```
data['Cluster_Kmeans_5'] = labels
data.head()
```

Out[52]:

	Index	interest_rate	credit	Gender	previous	duration	Churn	Cluster_Kmeans_3	Cluster_Kn
0	0	1.334	0	1	0	117	0	1	
1	1	0.767	0	0	1	274	1	1	
2	2	4.858	0	1	0	167	0	0	
3	3	4.120	0	0	0	686	1	0	
4	4	4.856	0	1	0	159	0	0	

Visualization

In [53]:
```
plt.scatter(x[y_kmeans==0,0], x[y_kmeans==0,1], s=100, c='blue',label='Cluster
plt.scatter(x[y_kmeans==1,0], x[y_kmeans==1,1], s=100, c='yellow',label='Clust
plt.scatter(x[y_kmeans==2,0], x[y_kmeans==2,1], s=100, c='green',label='Cluste
plt.scatter(x[y_kmeans==3,0], x[y_kmeans==3,1], s=100, c='black',label='Cluste
plt.scatter(x[y_kmeans==4,0], x[y_kmeans==4,1], s=100, c='purple',label='Clust
plt.show()
```



In [ ]:

In [ ]:

In [ ]: