

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

In [2]: df = pd.read_csv('bank+%281%29.csv')

In [3]: df.head()

Out[3]:
```

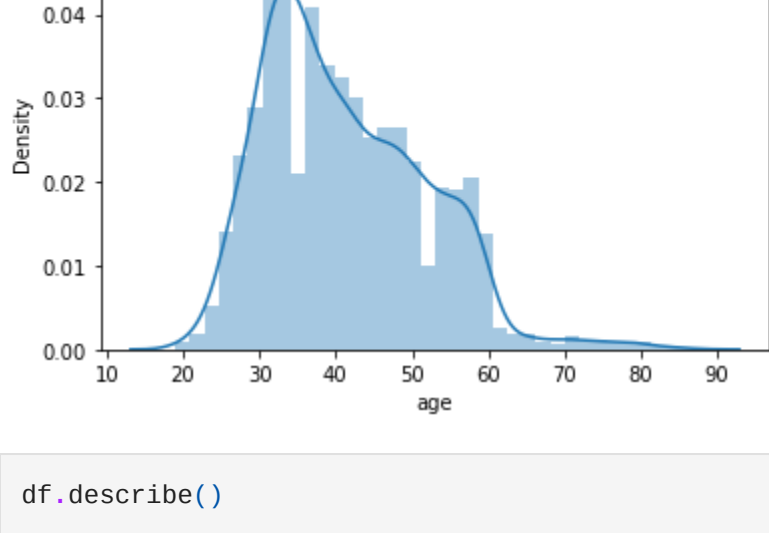
	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1	0	unknown	no
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339	4	failure	no
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330	1	failure	no
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1	0	unknown	no
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1	-1	0	unknown	no

```
In [4]: df.shape

Out[4]: (4521, 17)

In [5]: sns.distplot(df['age'])
plt.show()

C:\Users\Nilesh koli\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: 'distplot' is a deprecated function and will be removed in a future version. Please adapt your code to use either 'displot' (a figure-level function with similar flexibility) or 'histplot' (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)
```



A density plot showing the distribution of age. The x-axis is labeled 'age' and ranges from 10 to 90. The y-axis is labeled 'Density' and ranges from 0.00 to 0.05. The plot shows a unimodal distribution with a peak around age 35.

```
In [6]: df.describe()

Out[6]:
```

	age	balance	day	duration	campaign	pdays	previous
count	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000
mean	41.170095	1422.657819	15.915284	263.961292	2.793630	39.766645	0.542579
std	10.576211	3009.638142	8.247667	259.856633	3.109807	100.121124	1.693562
min	19.000000	-3313.000000	1.000000	4.000000	1.000000	-1.000000	0.000000
25%	33.000000	69.000000	9.000000	104.000000	1.000000	-1.000000	0.000000
50%	39.000000	444.000000	16.000000	185.000000	2.000000	-1.000000	0.000000
75%	49.000000	1480.000000	21.000000	329.000000	3.000000	-1.000000	0.000000
max	87.000000	71188.000000	31.000000	3025.000000	50.000000	871.000000	25.000000

```
In [10]: df['job'].value_counts(),df.shape

Out[10]:
```

(management	969
blue-collar	946
technician	768
admin.	478
services	417
retired	230
self-employed	183
entrepreneur	168
unemployed	128
housemaid	112
student	84
unknown	38

Name: job, dtype: int64, (4521, 17))

```
In [13]: df['marital'].value_counts()

Out[13]:
```

married	2797
single	1196
divorced	528

Name: marital, dtype: int64

```
In [14]: df['job'].value_counts().keys()

Out[14]:
```

Index(['management', 'blue-collar', 'technician', 'admin.', 'services', 'retired', 'self-employed', 'entrepreneur', 'unemployed', 'housemaid', 'student', 'unknown'], dtype='object')

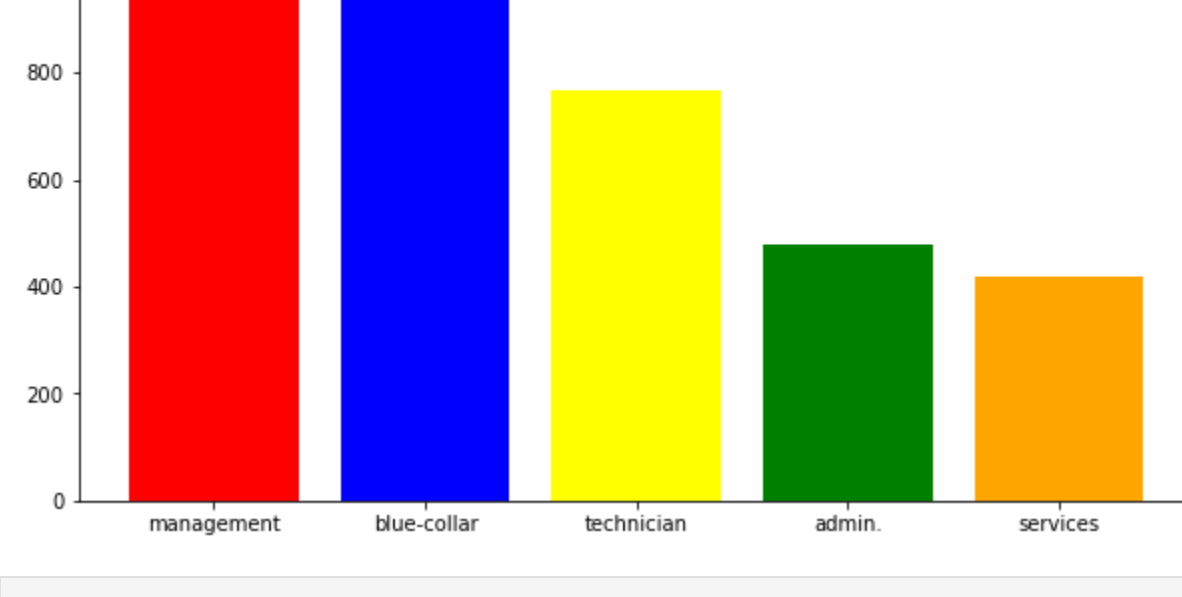
```
In [15]: df['job'].value_counts().values

Out[15]:
```

array([969, 946, 768, 478, 417, 230, 183, 168, 128, 112, 84, 38], dtype=int64)

```
In [26]: plt.figure(figsize=(10,5))
plt.bar(list(df['job'].value_counts().keys()[0:5]),list(df['job'].value_counts()[0:5]),color=['red','blue','yellow','green','orange'])
plt.show()

1000
800
600
400
200
0
management blue-collar technician admin. services
```



A bar chart showing the count of individuals for the top 5 job categories. The x-axis lists the job categories: management, blue-collar, technician, admin., and services. The y-axis represents the count, ranging from 0 to 1000. The bars are colored red, blue, yellow, green, and orange respectively.

```
In [27]: df['marital'].value_counts().keys()

Out[27]:
```

Index(['married', 'single', 'divorced'], dtype='object')

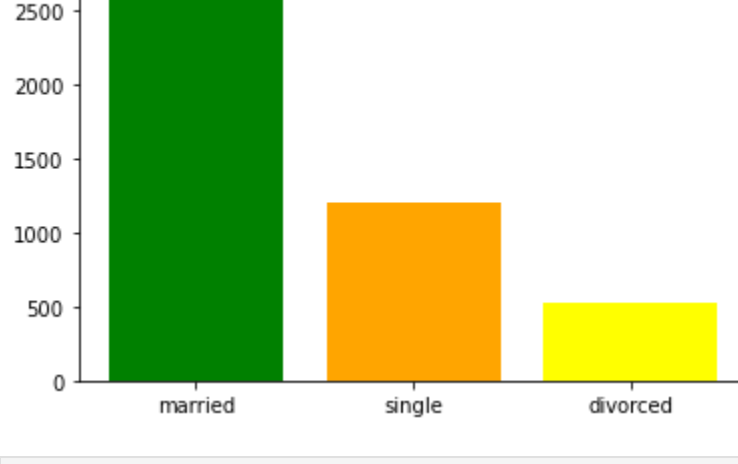
```
In [28]: df['marital'].value_counts().values

Out[28]:
```

array([2797, 1196, 528], dtype=int64)

```
In [35]: plt.bar(list(df['marital'].value_counts().keys()),list(df['marital'].value_counts()),color=['green','orange','yellow'])
plt.show()

2500
2000
1500
1000
500
0
married single divorced
```



A bar chart showing the count of individuals for each marital status. The x-axis lists the marital statuses: married, single, and divorced. The y-axis represents the count, ranging from 0 to 2500. The bars are colored green, orange, and yellow respectively.

```
In [36]: df.head()

Out[36]:
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1	0	unknown	no
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339	4	failure	no
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330	1	failure	no
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1	0	unknown	no
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1	-1	0	unknown	no

```
In [37]: df['education'].value_counts()

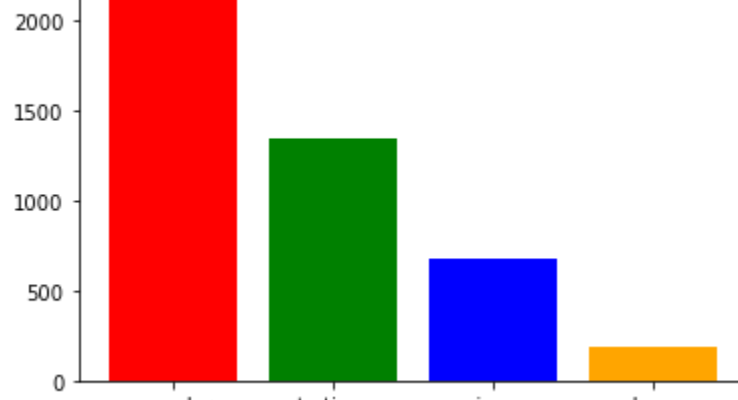
Out[37]:
```

secondary	2306
tertiary	1350
primary	678
unknown	187

Name: education, dtype: int64

```
In [39]: plt.bar(list(df['education'].value_counts().keys()),list(df['education'].value_counts()),color=['red','green','blue','orange'])
plt.show()

2000
1500
1000
500
0
secondary tertiary primary unknown
```



A bar chart showing the count of individuals for each education level. The x-axis lists the education levels: secondary, tertiary, primary, and unknown. The y-axis represents the count, ranging from 0 to 2000. The bars are colored red, green, blue, and orange respectively.

```
In [40]: df.head()

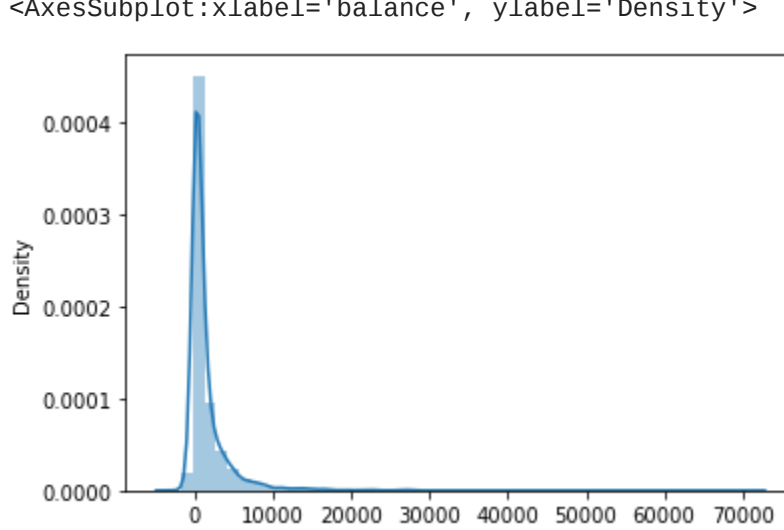
Out[40]:
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1	0	unknown	no
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339	4	failure	no
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330	1	failure	no
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1	0	unknown	no
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1	-1	0	unknown	no

```
In [41]: sns.distplot(df['balance'])

C:\Users\Nilesh koli\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: 'distplot' is a deprecated function and will be removed in a future version. Please adapt your code to use either 'displot' (a figure-level function with similar flexibility) or 'histplot' (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)
```

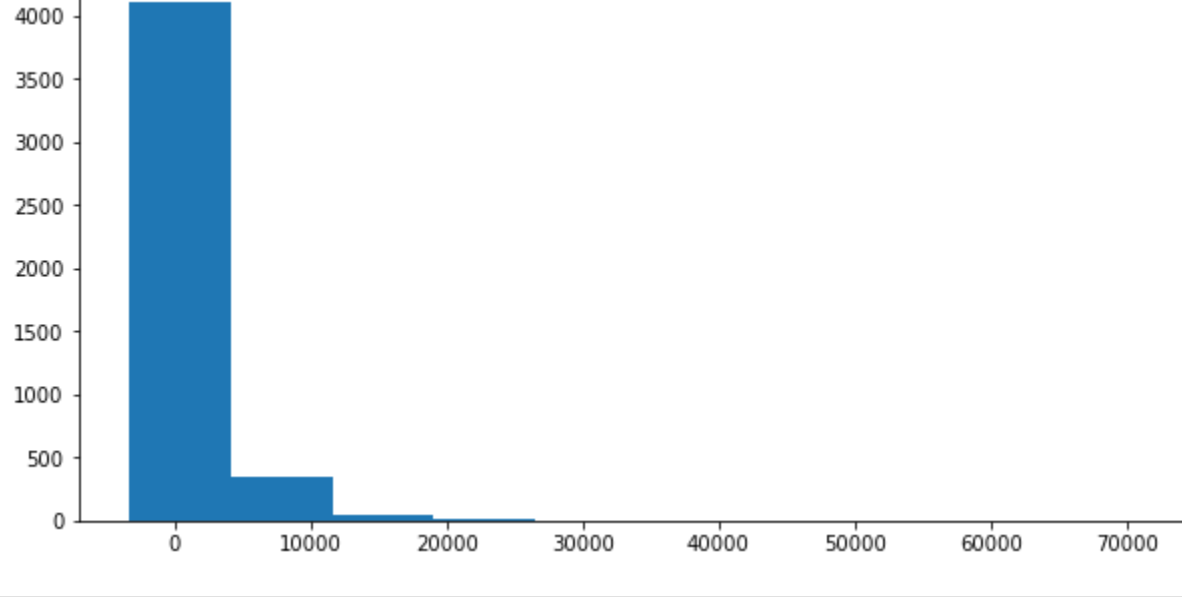
Out[41]: <AxesSubplot: xlabel='balance', ylabel='Density'>



A density plot showing the distribution of balance. The x-axis is labeled 'balance' and ranges from 0 to 70000. The y-axis is labeled 'Density' and ranges from 0.0000 to 0.0004. The plot shows a highly right-skewed distribution with a peak near zero.

```
In [46]: plt.figure(figsize=(10,5))
plt.hist(df['balance'])
plt.show()

4000
3500
3000
2500
2000
1500
1000
500
0
0 10000 20000 30000 40000 50000 60000 70000
```



A histogram showing the distribution of balance. The x-axis ranges from 0 to 70000, and the y-axis ranges from 0 to 4000. The distribution is highly right-skewed, with most values concentrated below 10000.

```
In [48]: df['loan'].value_counts(),df.shape

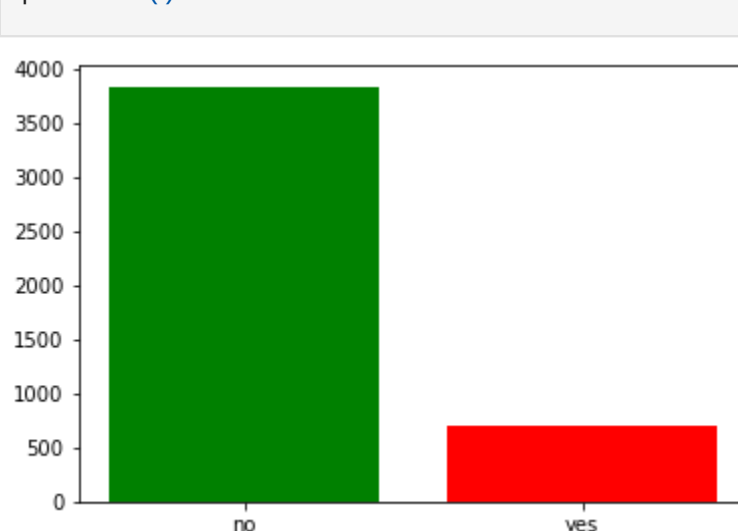
Out[48]:
```

(no	3839
yes	691

Name: loan, dtype: int64, (4521, 17))

```
In [50]: plt.bar(list(df['loan'].value_counts().keys()),list(df['loan'].value_counts()),color=['green','red'])
plt.show()

4000
3500
3000
2500
2000
1500
1000
500
0
no yes
```



A bar chart showing the count of individuals for each loan status. The x-axis lists the loan statuses: no and yes. The y-axis represents the count, ranging from 0 to 4000. The bars are colored green and red respectively.

```
In [51]: df.head()

Out[51]:
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1	0	unknown	no
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339	4	failure	no
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330	1	failure	no
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1	0	unknown	no
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1	-1	0	unknown	no

```
In [52]: x = df[['age']]

In [53]: y = df[['balance']]

In [54]: from sklearn.model_selection import train_test_split

In [55]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)

In [56]: from sklearn.ensemble import RandomForestRegressor

In [57]: model = RandomForestRegressor()
model

Out[57]: RandomForestRegressor()

In [58]: model.fit(x_train,y_train)

C:\python-input-58-4719cf73997a>1: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using.ravel().
model.fit(x_train,y_train)
RandomForestRegressor()

In [59]: y_pred = model.predict(x_test)

In [60]: y_test.head()

Out[60]:
```

balance
846
4366
3336
215
3039
8044
2143
4089
3462
360

```
In [61]: y_pred[:5]

Out[61]:
```

array([[1661.97596495, 1524.02834084, 1473.52962843, 1524.02834084, 1289.94669841])

```
In [62]: from sklearn.metrics import mean_squared_error

In [63]: print(mean_squared_error(y_test,y_pred))

18457944.18326088
```