

Assignment -2

Data Visualization and Pre-processing

Assignment Date	:	27 September 2022
Student Name	:	P.DHINESHKUMAR
Student Roll Number	:	912419104007
Maximum Marks	:	2 Marks

Task 1:

Download the dataset: [Dataset](#)

▾ Assignment-2

1. Download the dataset: [Dataset](#)

Task 2:

Question-1:

Loading the Churn_Modelling dataset

Solution:

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

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

Output:

1.Loading the Churn_Modelling dataset

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

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

Solution:

```
from google.colab import drive
drive.mount('/content/drive')
```

Output:

```
In [2]: from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive
```

Solution:

```
data = pd.read_csv("/content/Churn_Modelling.csv")
```

Output:

```
In [3]: data = pd.read_csv("/content/Churn_Modelling.csv")
```

Solution:

```
data.info()
```

Output:

```
In [4]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
#   Column             Non-Null Count  Dtype  
---  -
0   RowNumber           10000 non-null  int64  
1   CustomerId          10000 non-null  int64  
2   Surname             10000 non-null  object  
3   CreditScore          10000 non-null  int64  
4   Geography           10000 non-null  object  
5   Gender              10000 non-null  object  
6   Age                 10000 non-null  int64  
7   Tenure              10000 non-null  int64  
8   Balance             10000 non-null  float64 
9   NumOfProducts       10000 non-null  int64  
10  HasCrCard           10000 non-null  int64  
11  IsActiveMember      10000 non-null  int64  
12  EstimatedSalary     10000 non-null  float64 
13  Exited              10000 non-null  int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

Solution:

`data.head()`

Output:

```
In [5]: data.head()

Out[5]:
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

Solution:

```
data.tail()
```

Output:

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

Out[6]:	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
9995	9996	15606229	Obijaku	771	France	Male	39	5	0.00	2	1	0	96270.64	0
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	42085.58	1
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	38190.78	0

Solution:

```
data.shape
```

```
In [7]: data.shape
```

```
Out[7]: (10000, 14)
```

Task 3:

Question-2:

Visualization of Dataset

Univariate Analysis

- Distribution Plot

Solution:

```
penguins = sns.load_dataset("penguins")
sns.displot(penguins, x="flipper_length_mm")
```

Output:

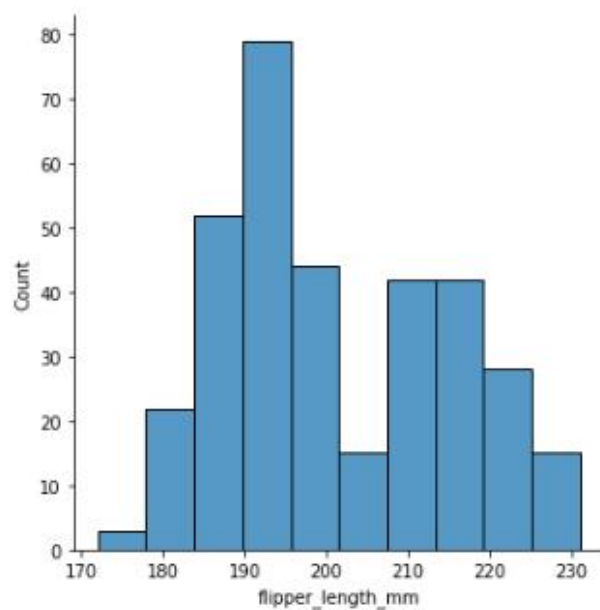
2.Vizualization of Dataset

Univariate Analysis

Distriution Plot

```
In [17]: penguins = sns.load_dataset("penguins")
sns.displot(penguins, x="flipper_length_mm")
```

```
Out[17]: <seaborn.axisgrid.FacetGrid at 0x7f3961965990>
```



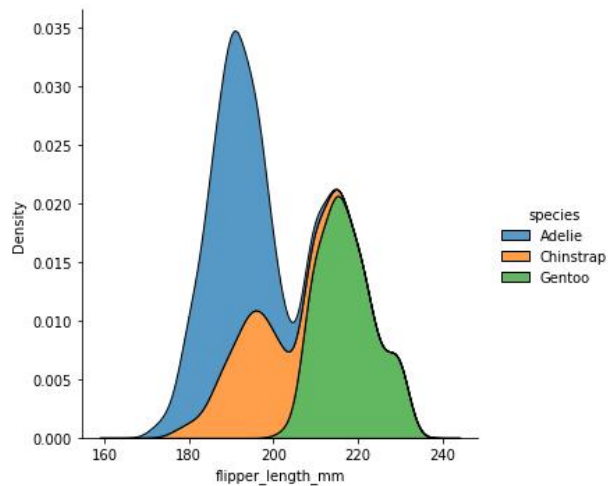
Solution:

```
sns.displot(penguins, x="flipper_length_mm", hue="species",
kind="kde", multiple="stack")
```

Output:

```
In [23]: sns.displot(penguins, x="flipper_length_mm", hue="species", kind="kde", multiple="stack")
```

```
Out[23]: <seaborn.axisgrid.FacetGrid at 0x7f395f278650>
```



● Histograms

Solution:

```
data['Geography'].value_counts()
```

Output:

```
In [12]: data['Geography'].value_counts()
```

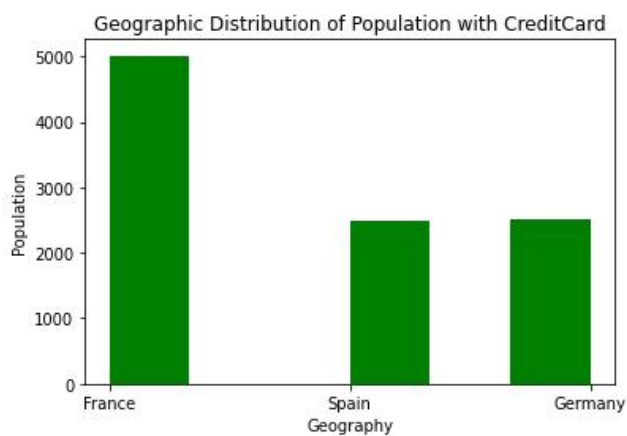
```
Out[12]: France      5014
Germany    2509
Spain      2477
Name: Geography, dtype: int64
```

Solution:

```
plt.hist(x=data.Geography, bins=6, color='blue')
plt.title("Geographic Distribution of Population with
CreditCard")
plt.xlabel("Geography")
plt.ylabel("Population")
plt.show()
```

Output:

```
In [21]: plt.hist(x=data.Geography, bins=6, color='green')
plt.title("Geographic Distribution of Population with CreditCard")
plt.xlabel("Geography")
plt.ylabel("Population")
plt.show()
```

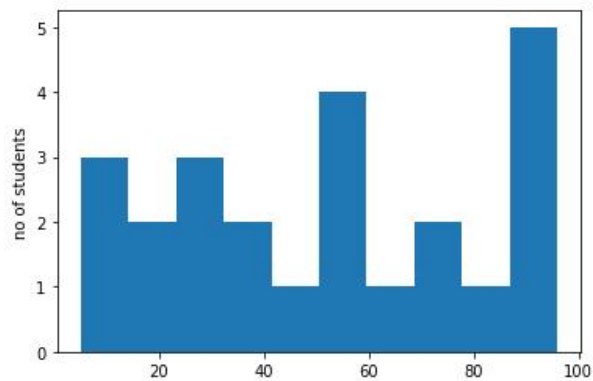


Solution:

```
fig, ax = plt.subplots(1, 1)
a = np.array([22, 87, 5, 43, 56, 73, 55, 54, 11, 20, 51, 5, 79, 31, 27, 63, 71, 90, 92, 95, 96,
32, 37, 40])
plt.hist(a)
ax.set_ylabel('no of students')
plt.show()
```

Output:

```
In [14]: fig,ax = plt.subplots(1,1)
a = np.array([22,87,5,43,56,73,55,54,11,20,51,5,79,31,27,63,71,90,92,95,96,32,37,40])
plt.hist(a)
ax.set_ylabel('no of students')
plt.show()
```



● Bar Plot

Solution:

```
data['Gender'].value_counts()
```

Output:

Bar Plot

```
In [15]: data['Gender'].value_counts()
```

```
Out[15]: Male      5457
Female    4543
Name: Gender, dtype: int64
```

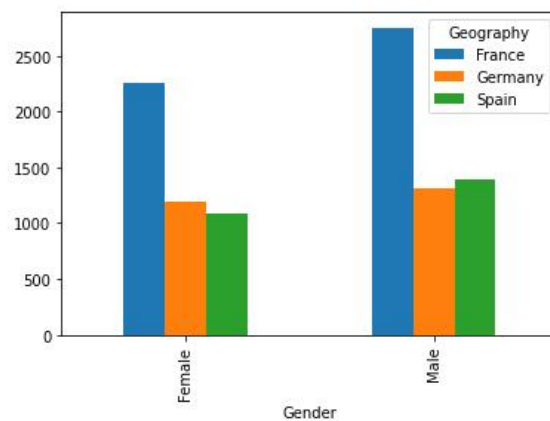

Solution:

```
pd.crosstab(data['Gender'],data['Geography']).plot(kind='bar')
```

Output:

```
In [ ]: pd.crosstab(data['Gender'],data['Geography']).plot(kind='bar')
```

```
Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4178be81d0>
```



Pie charts

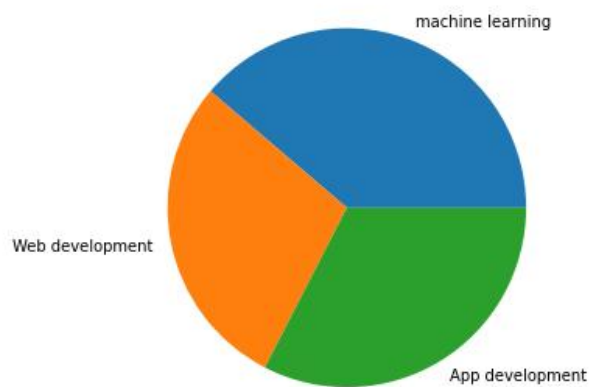
● Pie charts

Solution:

```
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
courses=['machine learning','Web development','App development']
students_enrolled=[50,37,42]
ax.pie(students_enrolled,labels=courses)
plt.show()
```

Output:

```
In [16]: fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
courses=['machine learning','Web development','App development']
students_enrolled=[50,37,42]
ax.pie(students_enrolled,labels=courses)
plt.show()
```



●Box plot

Solution:

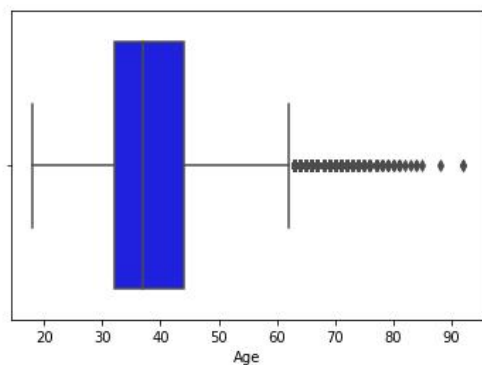
```
sns.boxplot(data['Age'],color=' blue' )
```

Output:

Box pLot

```
In [ ]: sns.boxplot(data["Age"],color='blue')
```

```
Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4178b5f650>
```



Bivariate Analysis

Solution:

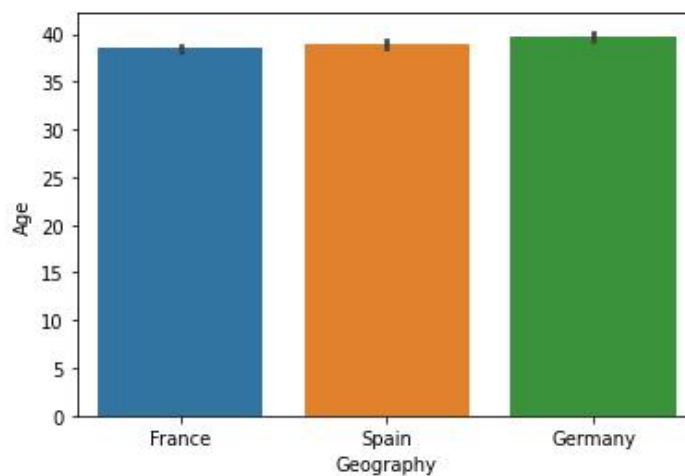
```
sns.barplot(data[ 'Geographgy' ], data["Age"])
```

Output:

Bivariate Analysis

```
In [ ]: sns.barplot(data['Geography'], data["Age"])
```

```
Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4178ae1190>
```

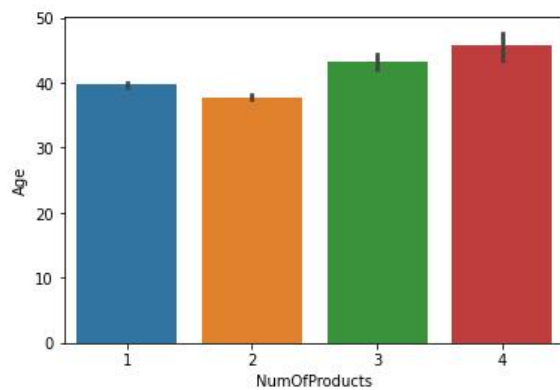


Solution:

```
sns.barplot(data["NumOfProducts"], data["Age"])
```

Output:

```
In [ ]: sns.barplot(data["NumOfProducts"],data["Age"])  
Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4178ac6f10>
```



Solution:

```
data['HasCrCard'].value_counts()
```

Output:

```
In [ ]: data['HasCrCard'].value_counts()  
Out[ ]: 1    7055  
        0    2945  
        Name: HasCrCard, dtype: int64
```

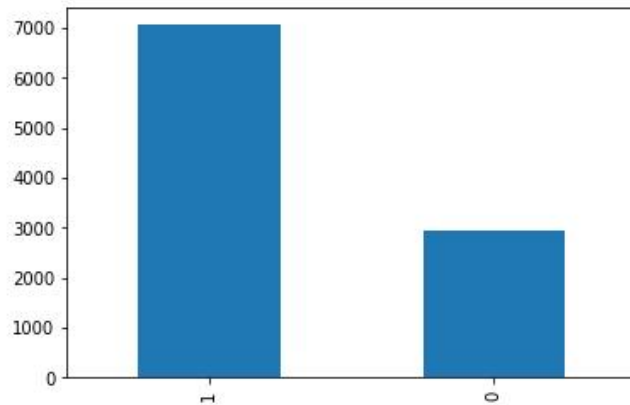
Solution:

```
data['HasCrCard'].value_counts().head(20).plot.bar()
```

Output:

```
In [ ]: data['HasCrCard'].value_counts().head(20).plot.bar()
```

```
Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f41789d3ad0>
```



• Line Chart

Solution:

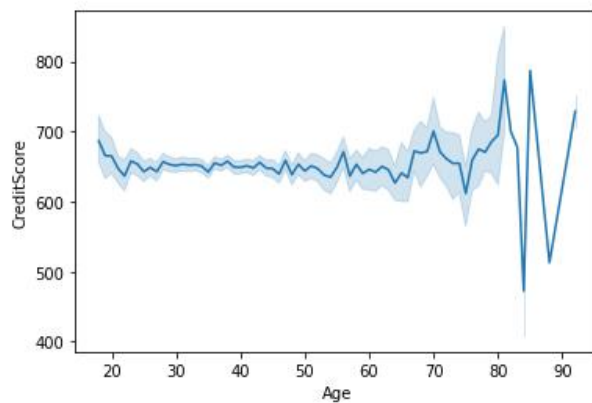
```
sns.lineplot(data['Age'], data['CreditScore'])
```

Output:

Line Chart

```
In [ ]: sns.lineplot(data['Age'], data['CreditScore'])
```

```
Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f41789a3290>
```



Multi-Variate Analysis

- Scatter Plot

Solution:

```
data['IsActiveMember'].value_counts()
```

Output:

Multi-Variate Analysis

Scatter Plot

```
In [ ]: data['IsActiveMember'].value_counts()
```

```
Out[ ]: 1    5151  
        0    4849  
        Name: IsActiveMember, dtype: int64
```

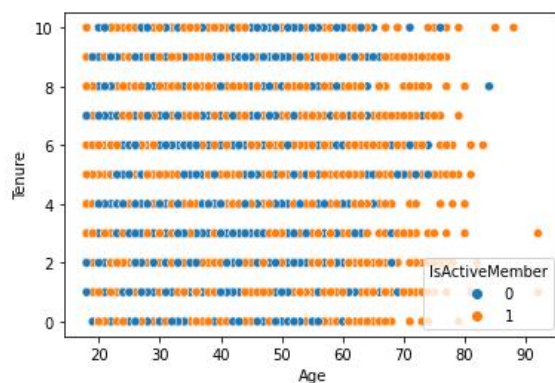
Solution:

```
sns.scatterplot(data['Age'], data['Tenure'],  
                hue=data['IsActiveMember'])
```

Output:

```
In [ ]: sns.scatterplot(data['Age'], data['Tenure'], hue=data['IsActiveMember'])
```

```
Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4178ae1f10>
```

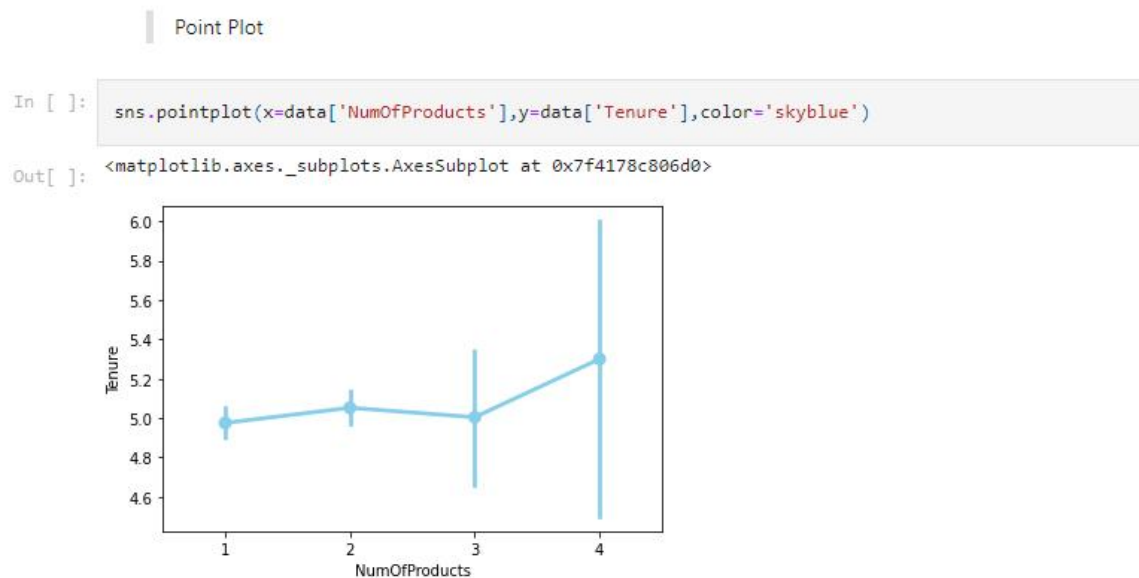


● Point Plot

Solution:

```
sns.pointplot(x=data['NumOfProducts'],y=data['Tenure'],color='skyblue')
```

Output:



● HeatMap

Solution:

```
data.head()
```

Output:

```
HeatMap
```

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

```
Out[ ]:
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

Solution:

```
data_cor = data.iloc[:,3:].corr()  
data_cor
```

Output:

```
In [ ]: data_cor = data.iloc[:,3:].corr()  
data_cor
```

```
Out[ ]:
```

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
CreditScore	1.000000	-0.003965	0.000842	0.006268	0.012238	-0.005458	0.025651	-0.001384	-0.027094
Age	-0.003965	1.000000	-0.009997	0.028308	-0.030680	-0.011721	0.085472	-0.007201	0.285323
Tenure	0.000842	-0.009997	1.000000	-0.012254	0.013444	0.022583	-0.028362	0.007784	-0.014001
Balance	0.006268	0.028308	-0.012254	1.000000	-0.304180	-0.014858	-0.010084	0.012797	0.118533
NumOfProducts	0.012238	-0.030680	0.013444	-0.304180	1.000000	0.003183	0.009612	0.014204	-0.047820
HasCrCard	-0.005458	-0.011721	0.022583	-0.014858	0.003183	1.000000	-0.011866	-0.009933	-0.007138
IsActiveMember	0.025651	0.085472	-0.028362	-0.010084	0.009612	-0.011866	1.000000	-0.011421	-0.156128
EstimatedSalary	-0.001384	-0.007201	0.007784	0.012797	0.014204	-0.009933	-0.011421	1.000000	0.012097
Exited	-0.027094	0.285323	-0.014001	0.118533	-0.047820	-0.007138	-0.156128	0.012097	1.000000

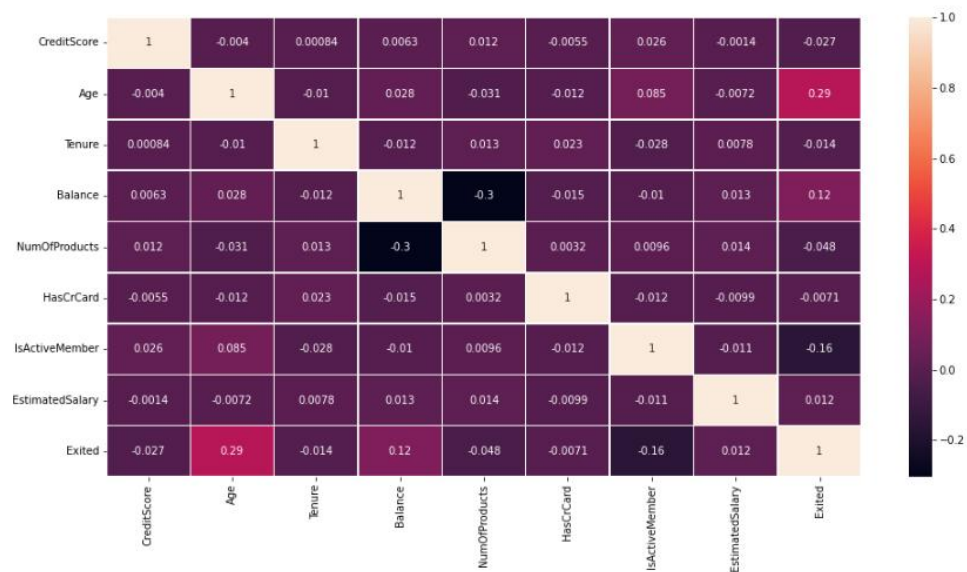
Solution:

```
plt.figure(figsize = (16,8))  
sns.heatmap(data_cor, linecolor='white', linewidth=0.5,  
annot=True)
```


Output:

```
In [ ]: plt.figure(figsize = (16,8))
sns.heatmap(data_cor,linecolor='white',linewidth=0.5, annot=True)
```

```
Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f41788d03d0>
```



● Pair Plot

Solution:

```
sns.pairplot(penguins)
```

Output:



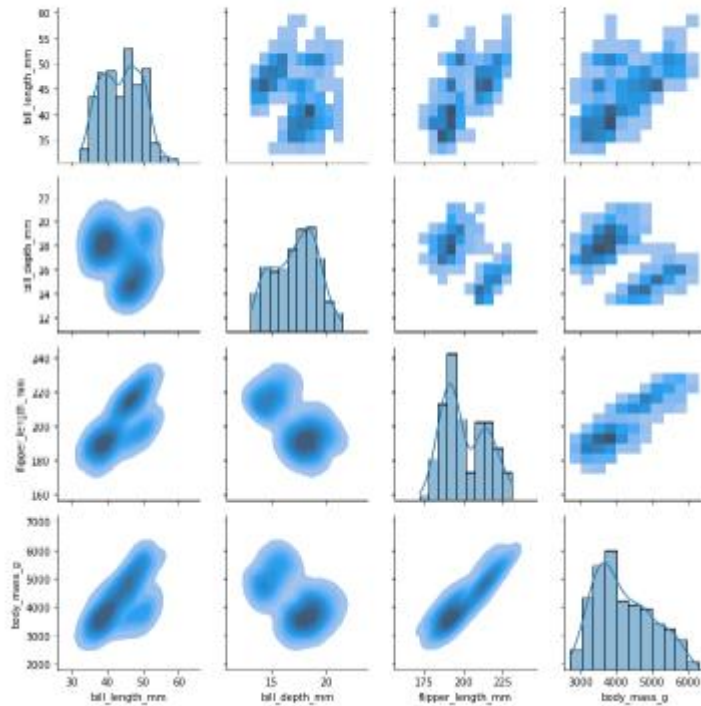
Solution:

```
g = sns.PairGrid(penguins)
g.map_upper(sns.histplot)
g.map_lower(sns.kdeplot, fill=True)
g.map_diag(sns.histplot, kde=True)
```

Output:

```
In [25]: g = sns.PairGrid(penguins)
g.map_upper(sns.histplot)
g.map_lower(sns.kdeplot, fill=True)
g.map_diag(sns.histplot, kde=True)
```

```
Out[25]: <seaborn.axisgrid.PairGrid at 0x7f995ca34190>
```



Task 4:

Question-3:

Descriptive Statistic Analysis

1. Mean
2. Medium
3. Mode
4. Standard Deviation
5. Variance

Solution:

```
data.describe().T
```

Output:

```
In [27]: data.describe().T
```

	count	mean	std	min	25%	50%	75%	max
RowNumber	10000.0	5.000500e+03	2886.895680	1.00	2500.75	5.000500e+03	7.500250e+03	10000.00
CustomerId	10000.0	1.569094e+07	71936.186123	15565701.00	15628528.25	1.569074e+07	1.575323e+07	15815690.00
CreditScore	10000.0	6.505288e+02	96.653299	350.00	584.00	6.520000e+02	7.180000e+02	850.00
Age	10000.0	3.892180e+01	10.487806	18.00	32.00	3.700000e+01	4.400000e+01	92.00
Tenure	10000.0	5.012800e+00	2.892174	0.00	3.00	5.000000e+00	7.000000e+00	10.00
Balance	10000.0	7.648589e+04	62397.405202	0.00	0.00	9.719854e+04	1.276442e+05	250898.09
NumOfProducts	10000.0	1.530200e+00	0.581654	1.00	1.00	1.000000e+00	2.000000e+00	4.00
HasCrCard	10000.0	7.055000e-01	0.455840	0.00	0.00	1.000000e+00	1.000000e+00	1.00
IsActiveMember	10000.0	5.151000e-01	0.499797	0.00	0.00	1.000000e+00	1.000000e+00	1.00
EstimatedSalary	10000.0	1.000902e+05	57510.492818	11.58	51002.11	1.001939e+05	1.493882e+05	199992.48
Exited	10000.0	2.037000e-01	0.402769	0.00	0.00	0.000000e+00	0.000000e+00	1.00

Solution:

```
data['Age'].mean()
```

Output:

```
In [35]: data['Age'].mean()

Out[35]: 38.9218
```

Solution:

```
data['Age'].median()
```

Output:

```
In [36]: data['Age'].median()
```

```
Out[36]: 37.0
```

Solution:

```
data['Age'].mode()
```

Output:

```
In [34]: data['Age'].mode()
```

```
Out[34]: 0    37  
dtype: int64
```

Solution:

```
data['EstimatedSalary'].mean()
```

Output:

```
In [33]: data['EstimatedSalary'].mean()
```

```
Out[33]: 100090.239881
```

Solution:

```
data['EstimatedSalary'].median(),)
```

Output:

```
In [32]: data['EstimatedSalary'].median()  
Out[32]: 100193.915
```

Solution:

```
data['EstimatedSalary'].mode()
```

Output:

```
In [31]: data['EstimatedSalary'].mode()  
Out[31]: 0    24924.92  
dtype: float64
```

Solution:

```
data['Balance'].mean()
```

Output:

```
In [30]: data['Balance'].mean()  
Out[30]: 76485.889288
```

Solution:

```
data['CreditScore'].std()
```

Output:

```
In [29]: data['CreditScore'].std()  
Out[29]: 96.65329873613035
```

Solution:

```
data['Tenure'].var()
```

Output:

```
In [28]: data['Tenure'].var()  
Out[28]: 8.364672627262726
```

Task 5:

Question-4:

Handling Missing Values

Solution:

```
data.isna().any()
```

Output:

4. Handling Missing Values

```
In [38]: data.isna().any()
```

```
Out[38]: RowNumber      False
         CustomerId     False
         Surname         False
         CreditScore     False
         Geography       False
         Gender           False
         Age              False
         Tenure           False
         Balance          False
         NumOfProducts   False
         HasCrCard        False
         IsActiveMember   False
         EstimatedSalary  False
         Exited           False
         dtype: bool
```

Solution:

```
data.dropna(inplace = True)
data.isnull().sum()
```

Output:

```
In [42]: data.dropna(inplace = True)
         data.isnull().sum()
```

```
Out[42]: RowNumber      0
         CustomerId     0
         Surname         0
         CreditScore     0
         Geography       0
         Gender           0
         Age              0
         Tenure           0
         Balance          0
         NumOfProducts   0
         HasCrCard        0
         IsActiveMember   0
         EstimatedSalary  0
         Exited           0
         dtype: int64
```


Solution:

```
data.isnull().sum()
```

Output:

```
In [37]: data.isnull().sum()
```

```
Out[37]: RowNumber      0  
CustomerId    0  
Surname       0  
CreditScore   0  
Geography     0  
Gender        0  
Age           0  
Tenure        0  
Balance       0  
NumOfProducts 0  
HasCrCard     0  
IsActiveMember 0  
EstimatedSalary 0  
Exited       0  
dtype: int64
```

Task 6:

Question-5:

Finding Outliers and Replacing Them

Solution:

```
outliers = data.quantile(q=(0.25,0.75))
```

Output:

5. Finding Outliers and Replacing Them

```
In [43]: outliers = data.quantile(q=(0.25,0.75))
```

Solution:

Outliers

Output:

In [44]:

```
outliers
```

Out[44]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0.25	2500.75	15628528.25	584.0	32.0	3.0	0.00	1.0	0.0	0.0	51002.1100	0.0
0.75	7500.25	15753233.75	718.0	44.0	7.0	127644.24	2.0	1.0	1.0	149388.2475	0.0

Solution:

```
iqr = outliers.loc[0.75]-outliers.loc[0.25]
```

Output:

In [47]:

```
iqr = outliers.loc[0.75]-outliers.loc[0.25]
```

Solution:

```
iqr[2:]
```

Output:

In [48]:

```
iqr[2:]
```

Out[48]:

```
CreditScore    134.0000
Age            12.0000
Tenure         4.0000
Balance        127644.2400
NumOfProducts   1.0000
HasCrCard       1.0000
IsActiveMember  1.0000
EstimatedSalary 98386.1375
Exited         0.0000
dtype: float64
```

Solution:

```
upper = outliers.loc[0.75] + 1.5 * iqr
```

Output:

```
In [49]: upper = outliers.loc[0.75] + 1.5 * iqr
```

Solution:

```
upper[2:]
```

Output:

```
In [51]: upper[2:]
Out[51]: CreditScore      919.00000
         Age              62.00000
         Tenure           13.00000
         Balance      319110.60000
         NumOfProducts      3.50000
         HasCrCard          2.50000
         IsActiveMember     2.50000
         EstimatedSalary  296967.45375
         Exited            0.00000
         dtype: float64
```

Solution:

```
`lower = outliers.loc[0.25] - 1.5 * iqr
```

Output:

```
In [50]: lower = outliers.loc[0.25] - 1.5 * iqr
```

Solution:

```
lower[2:]
```

Output:

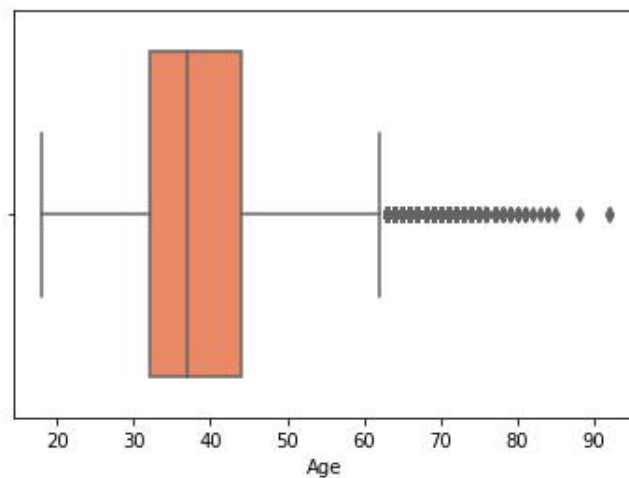
```
In [52]: lower[2:]
Out[52]: CreditScore      383.00000
Age                14.00000
Tenure             -3.00000
Balance           -191466.36000
NumOfProducts     -0.50000
HasCrCard          -1.50000
IsActiveMember     -1.50000
EstimatedSalary   -96577.09625
Exited              0.00000
dtype: float64
```

Solution:

```
sns.boxplot(data['Age'], color= 'Coral',)
```

Output:

```
In [53]: sns.boxplot(data['Age'], color= 'Coral',)
Out[53]: <matplotlib.axes._subplots.AxesSubplot at 0x7f395c0522d0>
```



Solution:

```
upper['Age']
```

Output:

```
In [54]: upper['Age']
```

```
Out[54]: 62.0
```

Solution:

```
data['Age'].mode()
```

Output:

```
In [55]: data['Age'].mode()
```

```
Out[55]: 0    37  
dtype: int64
```

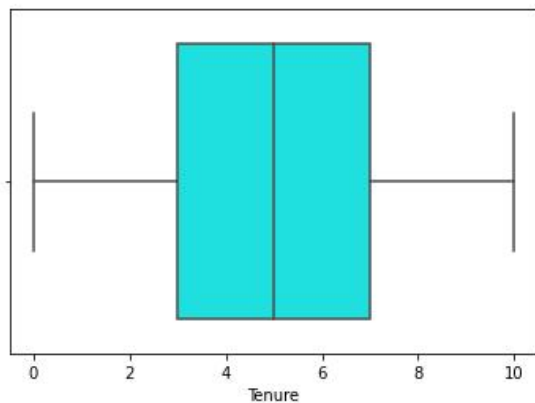
Solution:

```
sns.boxplot(data['Tenure'], color= 'cyan',)
```

Output:

```
In [56]: sns.boxplot(data['Tenure'], color= 'cyan',)
```

```
Out[56]: <matplotlib.axes._subplots.AxesSubplot at 0x7f395bfd3350>
```



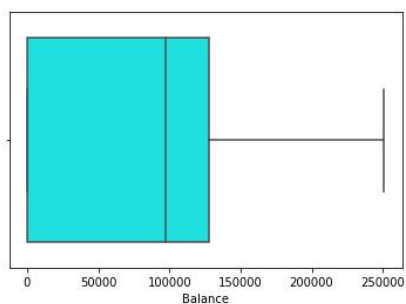
Solution:

```
sns.boxplot(data['Balance'], color= 'cyan',)
```

Output:

```
In [57]: sns.boxplot(data['Balance'], color= 'cyan',)
```

```
Out[57]: <matplotlib.axes._subplots.AxesSubplot at 0x7f395bfd8610>
```

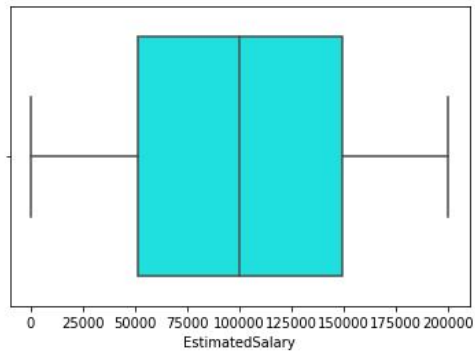


Solution:

```
sns.boxplot(data['Estimatedsalary'], color= 'cyan',)
```

Output:

```
In [58]: sns.boxplot(data['EstimatedSalary'], color= 'cyan',)
Out[58]: <matplotlib.axes._subplots.AxesSubplot at 0x7f395bf14350>
```

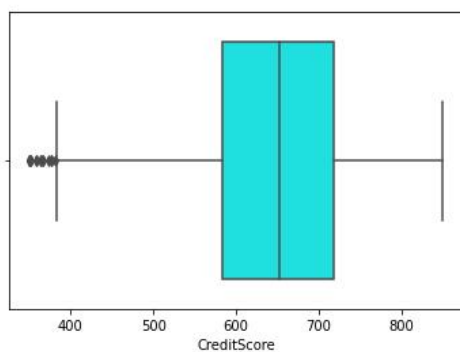


Solution:

```
sns.boxplot(data['CreditScore'], color= 'cyan',)
```

Output:

```
In [59]: sns.boxplot(data['CreditScore'], color= 'cyan',)
Out[59]: <matplotlib.axes._subplots.AxesSubplot at 0x7f395bea7d90>
```



Solution:

```
data['CreditScore'].mode()
```

Output:

```
In [60]: data['CreditScore'].mode()
```

```
Out[60]: 0    850  
dtype: int64
```

Solution:

```
lower['CreditScore']
```

Output:

```
In [61]: lower['CreditScore']
```

```
Out[61]: 383.0
```

Solution:

```
data["CreditScore"] =  
np.where(data["CreditScore"]<390, 850, data["CreditScore"])
```

Output:

```
In [ ]: data["CreditScore"] = np.where(data["CreditScore"]<390, 850, data["CreditScore"])
```

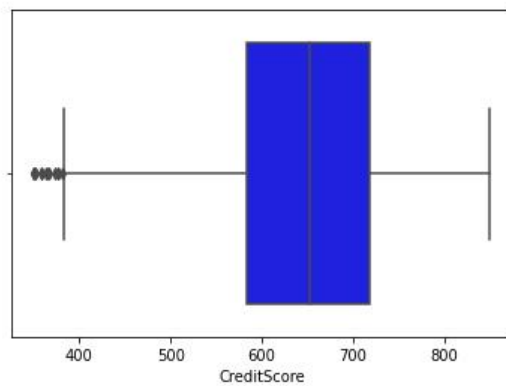
Solution:

```
sns.boxplot(data['CreditScore'], color= 'blue',)
```


Output:

```
In [62]: sns.boxplot(data['CreditScore'], color= 'blue',)
```

```
Out[62]: <matplotlib.axes._subplots.AxesSubplot at 0x7f395be0b2d0>
```



Task 7:

Question-6:

Checking for categorical columns and perform encoding

Solution:

```
data.info()
```

Output:

6. Checking for categorical columns and perform encoding

```
In [63]: data.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 10000 entries, 0 to 9999
Data columns (total 14 columns):
#   Column              Non-Null Count  Dtype  
---  -
0   RowNumber            10000 non-null  int64  
1   CustomerId           10000 non-null  int64  
2   Surname              10000 non-null  object  
3   CreditScore           10000 non-null  int64  
4   Geography            10000 non-null  object  
5   Gender               10000 non-null  object  
6   Age                  10000 non-null  int64  
7   Tenure               10000 non-null  int64  
8   Balance              10000 non-null  float64 
9   NumOfProducts        10000 non-null  int64  
10  HasCrCard            10000 non-null  int64  
11  IsActiveMember       10000 non-null  int64  
12  EstimatedSalary      10000 non-null  float64 
13  Exited               10000 non-null  int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

Solution:

```
data.dtypes.value_counts()
```

Output:

```
In [64]: data.dtypes.value_counts()

Out[64]: int64      9
         object     3
         float64    2
         dtype: int64
```

Solution:

```
# Encoding Categorical variables into numerical variables'
# Label Encoding

from sklearn.preprocessing import LabelEncode
label = LabelEncoder()
```

Output:

```
In [65]: # Encoding Categorical variables into numerical variables
# Label Encoding

from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
```

Solution:

```
data['Gender'] = label.fit_transform(data['Gender'])
data['Geography'] = label.fit_transform(data['Geography'])
```

Output:

```
In [66]: data['Gender'] = label.fit_transform(data['Gender'])
data['Geography'] = label.fit_transform(data['Geography'])
```

Solution:

```
data.head(8)
```

Output:

```
In [67]: data.head(8)
```

Out[67]:	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	0	0	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	2	0	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	0	0	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	0	0	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	2	0	43	2	125510.82	1	1	1	79084.10	0
5	6	15574012	Chu	645	2	1	44	8	113755.78	2	1	0	149756.71	1
6	7	15592531	Bartlett	822	0	1	50	7	0.00	2	1	1	10062.80	0
7	8	15656148	Obinna	376	1	0	29	4	115046.74	4	1	0	119346.88	1

Task 8:

Question-7:

Split the data into dependent and independent variables

Solution:

```
data_new = data.drop(['CustomerId', 'Surname', 'RowNumber'],  
axis = 1)  
data_new.info()
```

Output:

7.Split the data into dependent and independent variables

```
In [72]: data_new = data.drop(['CustomerId', 'Surname', 'RowNumber'], axis = 1)  
data_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
Int64Index: 10000 entries, 0 to 9999  
Data columns (total 11 columns):  
#   Column             Non-Null Count  Dtype  
---  ---             -  
0   CreditScore         10000 non-null  int64  
1   Geography           10000 non-null  int64  
2   Gender              10000 non-null  int64  
3   Age                 10000 non-null  int64  
4   Tenure              10000 non-null  int64  
5   Balance             10000 non-null  float64  
6   NumOfProducts       10000 non-null  int64  
7   HasCrCard           10000 non-null  int64  
8   IsActiveMember      10000 non-null  int64  
9   EstimatedSalary     10000 non-null  float64  
10  Exited              10000 non-null  int64  
dtypes: float64(2), int64(9)  
memory usage: 937.5 KB
```

Solution:

```
data_new.shape
```

Output:

```
In [73]: data_new.shape
```

```
Out[73]: (10000, 11)
```

Solution:

```
x = data_new.iloc[:,0:10]
y = data_new.iloc[:,10]

print(x.shape)
print(y.shape)

print(x.columns)
```

Output:

```
In [75]: x = data_new.iloc[:,0:10]
y = data_new.iloc[:,10]

print(x.shape)
print(y.shape)

print(x.columns)

(10000, 10)
(10000,)
Index(['CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'Balance',
      'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary'],
      dtype='object')
```

Solution:

```
x.head(8)
```

Output:

```
In [74]: x.head(8)
```

```
Out[74]:
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	619	0	0	42	2	0.00	1	1	1	101348.88
1	608	2	0	41	1	83807.86	1	0	1	112542.58
2	502	0	0	42	8	159660.80	3	1	0	113931.57
3	699	0	0	39	1	0.00	2	0	0	93826.63
4	850	2	0	43	2	125510.82	1	1	1	79084.10
5	645	2	1	44	8	113755.78	2	1	0	149756.71
6	822	0	1	50	7	0.00	2	1	1	10062.80
7	376	1	0	29	4	115046.74	4	1	0	119346.88

Task 9:

Question-8:

Split the data into training and testing

Solution:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size = 0.20, random_state = 0)

print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

Output:

8. Split the data into training and testing

```
In [76]: from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, random_state = 0)

print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)

(8000, 10)
(8000,)
(2000, 10)
(2000,)
```

Task 10:

Question-9:

Scale the independent variables

Solution:

```
from sklearn.preprocessing import StandardScaler  
ss = StandardScaler
```

Output:

9. Scale the independent variables

```
In [77]: from sklearn.preprocessing import StandardScaler  
         ss = StandardScaler
```

Solution:

```
from sklearn.preprocessing import StandardScaler  
  
sc = StandardScaler()  
x_train = sc.fit_transform(x_train)  
x_test = sc.fit_transform(x_test)  
  
x_train = pd.DataFrame(x_train)  
x_train.head()
```

Output:

```
In [79]: from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)

x_train = pd.DataFrame(x_train)
x_train.head()
```

```
Out[79]:
```

	0	1	2	3	4	5	6	7	8	9
0	0.169582	1.519198	-1.091687	-0.464608	0.006661	-1.215717	0.809503	0.642595	-1.032270	1.106432
1	-2.304559	0.313126	0.916013	0.301026	-1.377440	-0.006312	-0.921591	0.642595	0.968738	-0.748664
2	-1.191196	-0.892945	-1.091687	-0.943129	-1.031415	0.579935	-0.921591	0.642595	-1.032270	1.485335
3	0.035566	1.519198	0.916013	0.109617	0.006661	0.473128	-0.921591	0.642595	-1.032270	1.276528
4	2.056114	1.519198	-1.091687	1.736588	1.044737	0.810193	0.809503	0.642595	0.968738	0.558378