```
import pandas as pd
import numpy as np
students = [[86,"M","verygood"],[95,"F","Excellent"],[75,None,"Good"],[np.NaN,"M","Average"],[71,None,"Good"],[np.NaN,None,"\
```

```
dfstud = pd.DataFrame(students)
dfstud.columns = ['marks','gender','remark']
```

```
dfstud
```

	marks	gender	remark
0	86.0	M	verygood
1	95.0	F	Excellent
2	75.0	None	Good
3	NaN	M	Average
4	71.0	None	Good
5	NaN	None	Verygood
6	92.0	F	Verygood
7	99.0	M	Excellent

```
dfstud.isnull().values.sum()
```

```
5
```

```
X = dfstud.iloc[:,0:2].values
y = dfstud.iloc[:,2].values
type(X)
```

```
numpy.ndarray
```

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values = np.NaN,strategy ='mean')
X[:,0:1]= imputer.fit_transform(X[:,0:1])
print(X)
```

```
[[86.0 'M']
 [95.0 'F']
 [75.0 None]
 [86.33333333333333 'M']
 [71.0 None]
 [86.33333333333333 None]
 [92.0 'F']
 [99.0 'M']]
```

```
imputer = SimpleImputer(missing_values = np.NaN,strategy ='mean')
dfstud.marks = imputer.fit_transform(dfstud['marks'].values.reshape(-1,1))
dfstud
```

	marks	gender	remark
0	86.000000	M	verygood
1	95.000000	F	Excellent
2	75.000000	M	Good
3	86.333333	M	Average
4	71.000000	M	Good
5	86.333333	M	Verygood
6	92.000000	F	Verygood
7	99.000000	M	Excellent

```
imputer = SimpleImputer(missing_values = None,strategy ='most_frequent')
dfstud.gender = imputer.fit_transform(dfstud['gender'].values.reshape(-1,1))
dfstud
```

	marks	gender	remark
0	86.000000	M	verygood

1	95.000000	F	Excellent
2	75.000000	M	Good
3	86.333333	M	Average
4	71.000000	M	Good
5	86.333333	M	Verygood
6	92.000000	F	Verygood
7	99.000000	M	Excellent

Conclusion:

We can use sklearn.impute class SimpleImputer to impute missing Values for both numerical missing values, a strategy such as mean, median, most frequent and constants can be used. For categorical features a strategy such as the most frequent and constant can be used.

```
import pandas as pd
import numpy as np
```

```
df = pd.read_csv("/content/CountryAgeSalary.csv")
print(df)
```

```
   Country   Age   Salary Purchased
0   France  44.0  72000.0        No
1    Spain  27.0  48000.0       Yes
2  Germany  30.0  54000.0        No
3    Spain  38.0  61000.0        No
4  Germany  40.0      NaN       Yes
5   France  35.0  58000.0       Yes
6    Spain   NaN  52000.0        No
7   France  48.0  79000.0       Yes
8  Germany  50.0  83000.0        No
9   France  37.0  67000.0       Yes
```

```
dfstud.isnull().values.sum()
```

```
0
```

```
X = df.iloc[:,0:3].values
y = df.iloc[:,3].values
type(X)
```

```
numpy.ndarray
```

```
imputer = SimpleImputer(missing_values = np.NaN,strategy ='median')
X[:,1:3] = imputer.fit_transform(X[:,1:3])
X
```

```
array([['France', 44.0, 72000.0],
       ['Spain', 27.0, 48000.0],
       ['Germany', 30.0, 54000.0],
       ['Spain', 38.0, 61000.0],
       ['Germany', 40.0, 61000.0],
       ['France', 35.0, 58000.0],
       ['Spain', 38.0, 52000.0],
       ['France', 48.0, 79000.0],
       ['Germany', 50.0, 83000.0],
       ['France', 37.0, 67000.0]], dtype=object)
```

```
from sklearn.preprocessing import LabelEncoder
labelEncoder = LabelEncoder()
X[:,0] = labelEncoder.fit_transform(X[:,0])
print(X)
```

```
[[0 44.0 72000.0]
 [2 27.0 48000.0]
 [1 30.0 54000.0]
 [2 38.0 61000.0]
 [1 40.0 61000.0]
 [0 35.0 58000.0]
 [2 38.0 52000.0]
 [0 48.0 79000.0]
 [1 50.0 83000.0]
 [0 37.0 67000.0]]
```

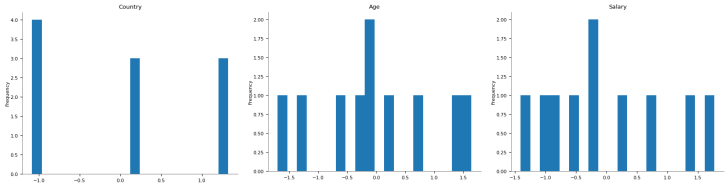
```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
columnTransformer = ColumnTransformer([("encoder",OneHotEncoder(),[0])],remainder = "passthrough")
z = np.array(columnTransformer.fit_transform(X),dtype = np.str)
z
```

```
<ipython-input-80-d1e9d0765856>:4: DeprecationWarning: `np.str` is a deprecated alias for the builtin `str`. To silence
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecatio
  z = np.array(columnTransformer.fit_transform(X),dtype = np.str)
array([['0.0', '1.0', '0.0', '0.0', '44.0', '72000.0'],
       ['1.0', '0.0', '0.0', '1.0', '27.0', '48000.0'],
       ['1.0', '0.0', '1.0', '0.0', '30.0', '54000.0'],
       ['1.0', '0.0', '0.0', '1.0', '38.0', '61000.0'],
       ['1.0', '0.0', '1.0', '0.0', '40.0', '61000.0'],
       ['0.0', '1.0', '0.0', '0.0', '35.0', '58000.0'],
       ['1.0', '0.0', '0.0', '1.0', '38.0', '52000.0'],
       ['0.0', '1.0', '0.0', '0.0', '48.0', '79000.0'],
       ['1.0', '0.0', '1.0', '0.0', '50.0', '83000.0'],
       ['0.0', '1.0', '0.0', '0.0', '37.0', '67000.0']], dtype='<U32')
```

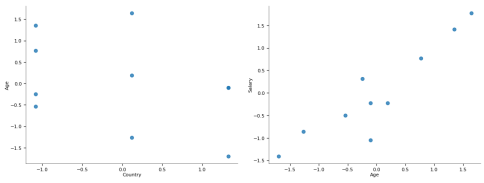
```
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
sc_x = sc_x.fit_transform(X)
sc_x = pd.DataFrame(data = sc_x, columns = ["Country",'Age','Salary'])
sc_x
```

	Country	Age	Salary
0	-1.083473	0.769734	0.772568
1	1.324244	-1.699225	-1.408800
2	0.120386	-1.263526	-0.863458
3	1.324244	-0.101663	-0.227226
4	0.120386	0.188803	-0.227226
5	-1.083473	-0.537362	-0.499897
6	1.324244	-0.101663	-1.045239
7	-1.083473	1.350666	1.408800
8	0.120386	1.641132	1.772361
9	-1.083473	-0.246896	0.318116

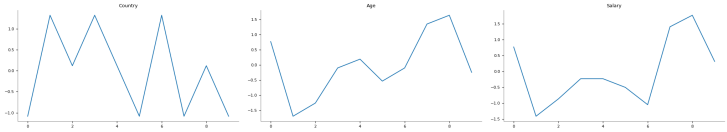
Distributions



2-d distributions



Values



X

```
array([[0, 44.0, 72000.0],
       [2, 27.0, 48000.0],
       [1, 30.0, 54000.0],
       [2, 38.0, 61000.0],
       [1, 40.0, 61000.0],
       [0, 35.0, 58000.0],
       [2, 38.0, 52000.0],
       [0, 48.0, 79000.0],
       [1, 50.0, 83000.0],
       [0, 37.0, 67000.0]], dtype=object)
```

Conclusion:

In this experiment we have successfully normalized the data using Standard Scaler.