

Data Preprocessing

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

1. Handling missing and duplicate data

```
df = pd.read_csv('../input/purchasedata/Data.csv')
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
10	France	37.0	67000.0	Yes

```
# Check if there are any duplicates
df.duplicated()
```

```
0      False
1      False
2      False
3      False
4      False
5      False
6      False
7      False
8      False
9      False
10     True
dtype: bool
```

```
# Drop Duplicates
df.drop_duplicates(inplace=True)
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

```
# Check if there is any missing data  
df.isnull()
```

	Country	Age	Salary	Purchased
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False
4	False	False	True	False
5	False	False	False	False
6	False	True	False	False
7	False	False	False	False
8	False	False	False	False
9	False	False	False	False

```
# Fill missing data with 0  
df.fillna(0)
```

	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	0.0	Yes
5	France	35.0	58000.0	Yes
6	Spain	0.0	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
# Or Drop entries with missing data  
df.dropna()
```

	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
5	France	35.0	58000.0	Yes
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
# Or fill the missing data with the mean of corresponding column  
avg_age = df['Age'].mean()  
avg_salary = df['Salary'].mean()  
print(avg_age)  
print(avg_salary)
```

38.77777777777778

63777.77777777778

```
df['Age'].replace(np.nan, avg_age, inplace = True)
df['Salary'].replace(np.nan, avg_salary, inplace = True)
df
```

	Country	Age	Salary	Purchased
0	France	44.000000	72000.000000	No
1	Spain	27.000000	48000.000000	Yes
2	Germany	30.000000	54000.000000	No
3	Spain	38.000000	61000.000000	No
4	Germany	40.000000	63777.777778	Yes
5	France	35.000000	58000.000000	Yes
6	Spain	38.777778	52000.000000	No
7	France	48.000000	79000.000000	Yes
8	Germany	50.000000	83000.000000	No
9	France	37.000000	67000.000000	Yes

Using Scikit-Learn

```

df2 = pd.read_csv('../input/purchasedata/Data.csv')
df2.drop_duplicates(inplace = True)
X = df2.iloc[:, :-1].values
Y = df2.iloc[:, -1].values
print(X)
print(Y)
df2

```

```

[['France' 44.0 72000.0]
 ['Spain' 27.0 48000.0]
 ['Germany' 30.0 54000.0]
 ['Spain' 38.0 61000.0]
 ['Germany' 40.0 nan]
 ['France' 35.0 58000.0]
 ['Spain' nan 52000.0]
 ['France' 48.0 79000.0]
 ['Germany' 50.0 83000.0]
 ['France' 37.0 67000.0]]
['No' 'Yes' 'No' 'No' 'Yes' 'Yes' 'No' 'Yes' 'No' 'Yes']

```

	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

```
# Using Scikit-Learn to deal with missing data
from sklearn.impute import SimpleImputer
imp = SimpleImputer(missing_values = np.nan, strategy = 'mean')
imp.fit(X[:,1:3])
X[:, 1:3] = imp.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, 63777.77777777778],
      ['France', 35.0, 58000.0],
      ['Spain', 38.77777777777778, 52000.0],
      ['France', 48.0, 79000.0],
      ['Germany', 50.0, 83000.0],
      ['France', 37.0, 67000.0]], dtype=object)
```

2.Feature scaling(normalisation)

```
from sklearn.preprocessing import StandardScaler          #MinMa
xScaler can also be used
sc = StandardScaler()
X[:,1:] = sc.fit_transform(X[:,1:])
X
```

```
array([[ 'France', 0.758874361590019, 0.7494732544921677],
      [ 'Spain', -1.7115038793306814, -1.438178407268753
1],
      [ 'Germany', -1.2755547779917342, -0.891265491828522
9],
      [ 'Spain', -0.1130238410878753, -0.253200423814921],
      [ 'Germany', 0.17760889313808945, 6.632191985654332e
-16],
      [ 'France', -0.5489729424268225, -0.526656881535036
1],
      [ 'Spain', 0.0, -1.0735697969752662],
      [ 'France', 1.3401398300419485, 1.3875383225057696],
      [ 'Germany', 1.6307725642679132, 1.752146932799256
5],
      [ 'France', -0.2583402082008577, 0.2937124916253091
6]], dtype=object)
```

3. Handling categorical data

Embed cell *Using independent variable*

```
dummy1 = pd.get_dummies(df['Country'])  
dummy1
```

	France	Germany	Spain
0	1	0	0
1	0	0	1
2	0	1	0
3	0	0	1
4	0	1	0
5	1	0	0
6	0	0	1
7	1	0	0
8	0	1	0
9	1	0	0

```
df = pd.concat([df,dummy1], axis=1)  
df
```

	Country	Age	Salary	Purchased	France	Germany	Spain
0	France	44.000000	72000.000000	No	1	0	0
1	Spain	27.000000	48000.000000	Yes	0	0	1
2	Germany	30.000000	54000.000000	No	0	1	0
3	Spain	38.000000	61000.000000	No	0	0	1
4	Germany	40.000000	63777.777778	Yes	0	1	0
5	France	35.000000	58000.000000	Yes	1	0	0
6	Spain	38.777778	52000.000000	No	0	0	1
7	France	48.000000	79000.000000	Yes	1	0	0
8	Germany	50.000000	83000.000000	No	0	1	0
9	France	37.000000	67000.000000	Yes	1	0	0

```
dummy = df.iloc[:,4:].values  
dummy
```

```
array([[1, 0, 0],  
       [0, 0, 1],  
       [0, 1, 0],  
       [0, 0, 1],  
       [0, 1, 0],  
       [1, 0, 0],  
       [0, 0, 1],  
       [1, 0, 0],  
       [0, 1, 0],  
       [1, 0, 0]], dtype=uint8)
```

```
#Encoding dependent variable  
dummy2 = pd.get_dummies(df['Purchased'])  
dummy2
```

	No	Yes
0	1	0
1	0	1
2	1	0
3	1	0
4	0	1
5	0	1
6	1	0
7	0	1
8	1	0
9	0	1

Using Scikit-Learn

```
#Encoding independent variable
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], remainder='passthrough')
X = np.array(ct.fit_transform(X))
X
```

:

```
array([[1.0, 0.0, 0.0, 0.758874361590019, 0.74947325449216
77],
      [0.0, 0.0, 1.0, -1.7115038793306814, -1.43817840726
87531],
      [0.0, 1.0, 0.0, -1.2755547779917342, -0.89126549182
85229],
      [0.0, 0.0, 1.0, -0.1130238410878753, -0.25320042381
4921],
      [0.0, 1.0, 0.0, 0.17760889313808945, 6.632191985654
332e-16],
      [1.0, 0.0, 0.0, -0.5489729424268225, -0.52665688153
50361],
      [0.0, 0.0, 1.0, 0.0, -1.0735697969752662],
      [1.0, 0.0, 0.0, 1.3401398300419485, 1.3875383225057
696],
      [0.0, 1.0, 0.0, 1.6307725642679132, 1.7521469327992
565],
      [1.0, 0.0, 0.0, -0.2583402082008577, 0.293712491625
30916]],
      dtype=object)
```

```
#Encoding Dependent variable
```

```
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()
```

```
Y = le.fit_transform(Y)
```

```
Y
```

```
array([0, 1, 0, 0, 1, 1, 0, 1, 0, 1])
```

```
print(X)
```

```
print(Y)
```

```
[[1.0 0.0 0.0 0.758874361590019 0.7494732544921677]
 [0.0 0.0 1.0 -1.7115038793306814 -1.4381784072687531]
 [0.0 1.0 0.0 -1.2755547779917342 -0.8912654918285229]
 [0.0 0.0 1.0 -0.1130238410878753 -0.253200423814921]
 [0.0 1.0 0.0 0.17760889313808945 6.632191985654332e-16]
 [1.0 0.0 0.0 -0.5489729424268225 -0.5266568815350361]
 [0.0 0.0 1.0 0.0 -1.0735697969752662]
 [1.0 0.0 0.0 1.3401398300419485 1.3875383225057696]
 [0.0 1.0 0.0 1.6307725642679132 1.7521469327992565]
 [1.0 0.0 0.0 -0.2583402082008577 0.29371249162530916]]
[0 1 0 0 1 1 0 1 0 1]
```

4. Splitting the data into training and test sets

```
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 = 0)
print(X_train)
print(y_train)
```

```
[[0.0  1.0  0.0  0.17760889313808945  6.632191985654332e-16]
 [1.0  0.0  0.0 -0.2583402082008577  0.29371249162530916]
 [0.0  0.0  1.0 -1.7115038793306814 -1.4381784072687531]
 [0.0  0.0  1.0  0.0 -1.0735697969752662]
 [1.0  0.0  0.0  1.3401398300419485  1.3875383225057696]
 [0.0  0.0  1.0 -0.1130238410878753 -0.253200423814921]
 [1.0  0.0  0.0  0.758874361590019  0.7494732544921677]
 [1.0  0.0  0.0 -0.5489729424268225 -0.5266568815350361]]
[1 1 1 0 1 0 0 1]
```

```
print(X_test, y_test)
```

```
[[0.0  1.0  0.0 -1.2755547779917342 -0.8912654918285229]
 [0.0  1.0  0.0  1.6307725642679132  1.7521469327992565]] [0
0]
```