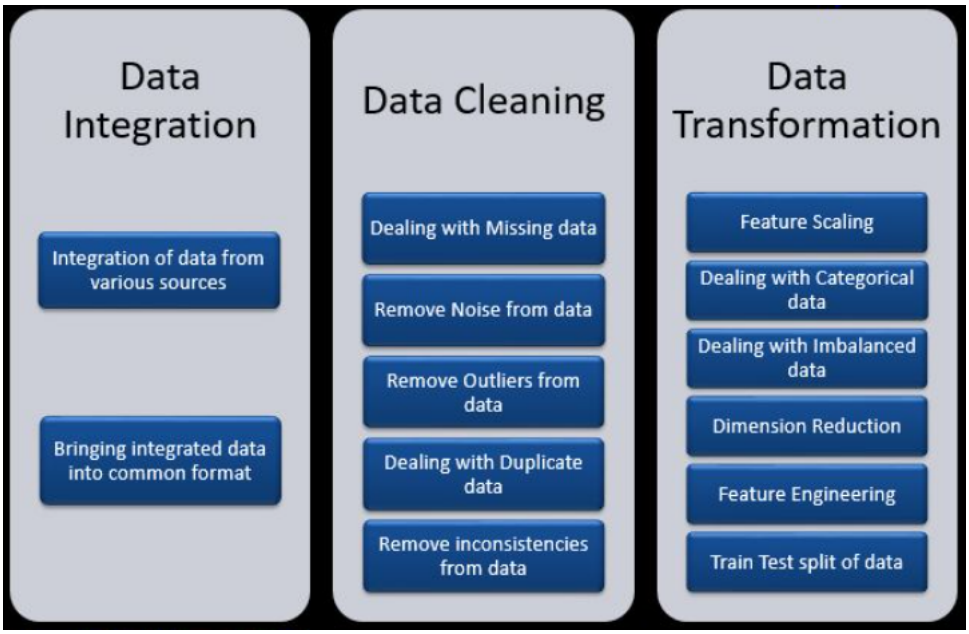
# Data Preprocessing

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.




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

```
df = pd.read_csv("/content/drive/MyDrive/Datasets/nba.csv")
```

```
df.sample(10)
```

| | Name | Team | Number | Position | Age | Height | Weight | College | Salary |
|---|---|---|---|---|---|---|---|---|---|
| **289** | Jordan Hamilton | New Orleans Pelicans | 25.0 | SG | 25.0 | 6-7 | 220.0 | Texas | 1015421.0 |
| **84** | Shaun Livingston | Golden State Warriors | 34.0 | PG | 30.0 | 6-7 | 192.0 | NaN | 5543725.0 |
| **369** | Bradley Beal | Washington Wizards | 3.0 | SG | 22.0 | 6-5 | 207.0 | Florida | 5694674.0 |
| **116** | D'Angelo Russell | Los Angeles Lakers | 1.0 | PG | 20.0 | 6-5 | 195.0 | Ohio State | 5103120.0 |
| **363** | Victor Oladipo | Orlando Magic | 5.0 | SG | 24.0 | 6-4 | 210.0 | Indiana | 5192520.0 |
| **96** | Blake Griffin | Los Angeles Clippers | 32.0 | PF | 27.0 | 6-10 | 251.0 | Oklahoma | 18907726.0 |
| **254** | Donatas Motiejunas | Houston Rockets | 20.0 | PF | 25.0 | 7-0 | 222.0 | NaN | 2288205.0 |
| **408** | Damjan Rudez | Minnesota Timberwolves | 10.0 | SF | 29.0 | 6-9 | 230.0 | NaN | 1149500.0 |
| **365** | Jason Smith | Orlando Magic | 14.0 | PF | 30.0 | 7-0 | 240.0 | Colorado State | 4300000.0 |

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 458 entries, 0 to 457
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Name      457 non-null    object
 1   Team      457 non-null    object
 2   Number    457 non-null    float64
 3   Position  457 non-null    object
 4   Age       457 non-null    float64
 5   Height    457 non-null    object
```

```
 6   Weight    457 non-null    float64
 7   College   373 non-null    object
 8   Salary    446 non-null    float64
dtypes: float64(4), object(5)
memory usage: 32.3+ KB
```

```
df.describe()
```

| | Number | Age | Weight | Salary |
|---|---|---|---|---|
| **count** | 457.000000 | 457.000000 | 457.000000 | 4.460000e+02 |
| **mean** | 17.678337 | 26.938731 | 221.522976 | 4.842684e+06 |
| **std** | 15.966090 | 4.404016 | 26.368343 | 5.229238e+06 |
| **min** | 0.000000 | 19.000000 | 161.000000 | 3.088800e+04 |
| **25%** | 5.000000 | 24.000000 | 200.000000 | 1.044792e+06 |
| **50%** | 13.000000 | 26.000000 | 220.000000 | 2.839073e+06 |
| **75%** | 25.000000 | 30.000000 | 240.000000 | 6.500000e+06 |
| **max** | 99.000000 | 40.000000 | 307.000000 | 2.500000e+07 |

```
df.describe(include=['object'])
```

| | Name | Team | Position | Height | College |
|---|---|---|---|---|---|
| **count** | 457 | 457 | 457 | 457 | 373 |
| **unique** | 457 | 30 | 5 | 18 | 118 |
| **top** | Avery Bradley | New Orleans Pelicans | SG | 6-9 | Kentucky |
| **freq** | 1 | 19 | 102 | 59 | 22 |

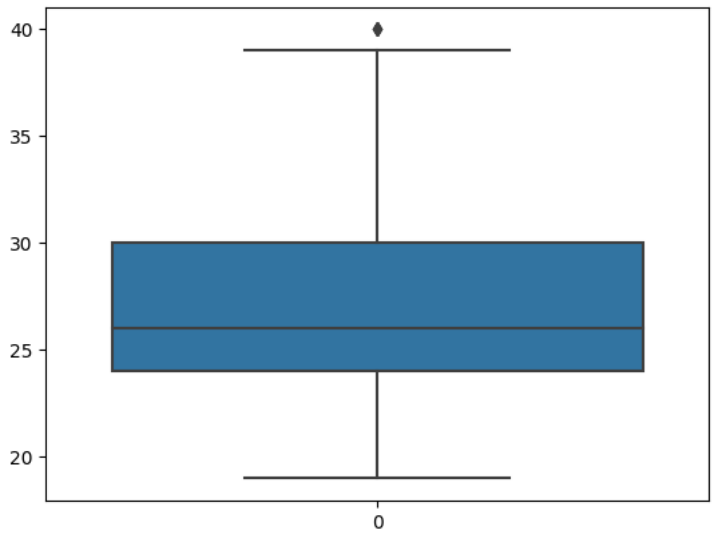
```
df.isna().sum()
```

```
Name         1
Team         1
Number       1
Position     1
Age          1
Height       1
Weight       1
College     85
Salary      12
dtype: int64
```

### Outlier Detection and Removal

```
# Detecting Outliers through Box plot
sns.boxplot(df['Age'])
```

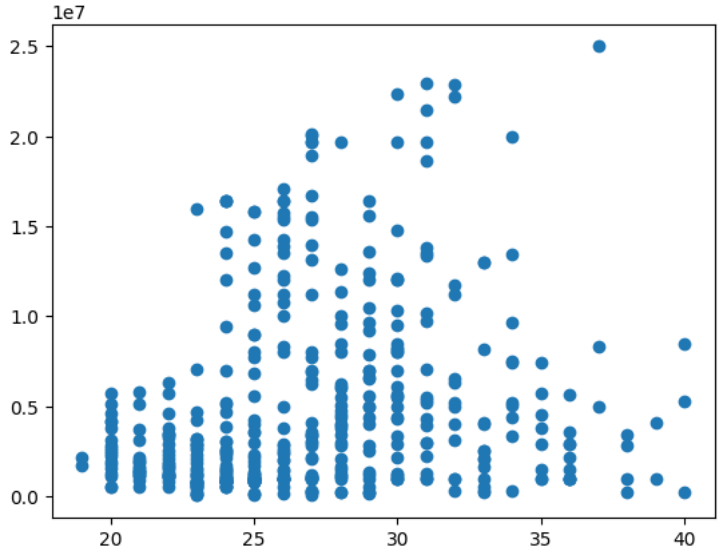
```
<Axes: >
```




```
print(np.where(df['Age']>39))
```

```
(array([298, 304, 400]),)
```

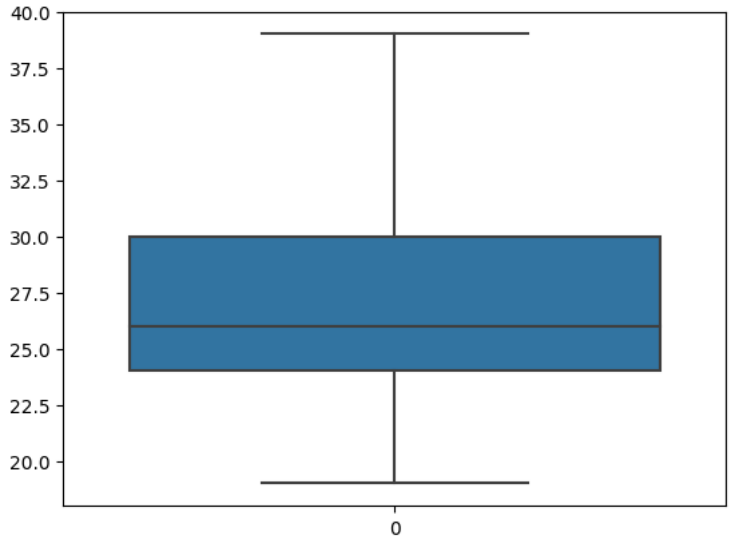
```
# Detecting Outliers through Scatter Plot
plt.scatter(df['Age'], df['Salary'])
plt.show()
```




```
df.drop([298, 304, 400], inplace=True)
```

```
sns.boxplot(df['Age'])
```

```
<Axes: >
```




**Handling Missing Values: 1. Deleting only the rows/columns with missing values**

```
df1 = df.copy()
df1 = df1.dropna(axis=0)
df1.isna().sum()
```

```
Name        0
Team        0
Number      0
Position    0
Age         0
Height      0
Weight      0
College     0
Salary      0
dtype: int64
```

```
df1.shape
```

**Handling Missing Values: 2. Imputing missing values with Mean/Median/Mode**

```
df2 = df.copy()
df2['Salary']=df2['Salary'].fillna(df2['Salary'].mean())
df2.dropna(axis=0,inplace=True, how='any')
df2.isna().sum()
```

```
Name        0
Team        0
Number      0
Position    0
Age         0
Height      0
Weight      0
College     0
Salary      0
dtype: int64
```

```
df2.shape
```

### Handling Missing Values: 3. Simple Imputer

```
from sklearn.impute import SimpleImputer
df3 = df.copy()
df3['Salary_Missing'] = df3['Salary'].isnull()
df3.dropna(axis=0,inplace=True, how='any', subset=['Name', 'Team', 'Position', 'Height', 'College'])

my_imputer = SimpleImputer(missing_values=np.NaN, strategy = 'mean') # the imputation strategy can be 'mean'/'median'/'most_frequent'/'cc
new_data = my_imputer.fit_transform(df3['Salary'].values.reshape(-1,1))[:,0]
df3.Salary = new_data

df3.isnull().sum()
```

```
Name              0
Team              0
Number            0
Position          0
Age               0
Height            0
Weight            0
College           0
Salary            0
Salary_Missing    0
dtype: int64
```

```
df3.sample(10)
```

### Formatting data

```
df3['Height_In_Inches'] = df3['Height'].apply(lambda x: 12*int(x[:1]) + int(x[2:]))
df3.drop('Height', inplace=True, axis=1)
```

```
df3.sample(5)
```

| | Name | Team | Number | Position | Age | Weight | College | Salary | Salary_Missing | Height_In_Inches |
|---|---|---|---|---|---|---|---|---|---|---|
| **189** | Tobias Harris | Detroit Pistons | 34.0 | SF | 23.0 | 235.0 | Tennessee | 16000000.0 | False | 81 |
| **242** | Trevor Ariza | Houston Rockets | 1.0 | SF | 30.0 | 215.0 | UCLA | 8193030.0 | False | 80 |
| **101** | Paul Pierce | Los Angeles Clippers | 34.0 | SF | 38.0 | 235.0 | Kansas | 3376000.0 | False | 79 |
| **428** | Al-Farouq Aminu | Portland Trail Blazers | 8.0 | SF | 25.0 | 215.0 | Wake Forest | 8042895.0 | False | 81 |

### Removing Duplicates

```
df4 = df3.copy()
df4.duplicated().sum()
```

```
0
```

```
# Duplicating the instances
df4 = pd.concat([df4]*2, ignore_index=True)
```

```
df4.duplicated().sum()
```

```
371
```

```
df4[df4.duplicated()]
```

| | Name | Team | Number | Position | Age | Weight | College | Salary | Salary_Missing | Height_In_Inches |
|---|---|---|---|---|---|---|---|---|---|---|
| **371** | Avery Bradley | Boston Celtics | 0.0 | PG | 25.0 | 180.0 | Texas | 7.730337e+06 | False | 74 |
| **372** | Jae Crowder | Boston Celtics | 99.0 | SF | 25.0 | 235.0 | Marquette | 6.796117e+06 | False | 78 |
| **373** | John Holland | Boston Celtics | 30.0 | SG | 27.0 | 205.0 | Boston University | 4.630642e+06 | True | 77 |
| **374** | R.J. Hunter | Boston Celtics | 28.0 | SG | 22.0 | 185.0 | Georgia State | 1.148640e+06 | False | 77 |
| **375** | Jordan Mickey | Boston Celtics | 55.0 | PF | 21.0 | 235.0 | LSU | 1.170960e+06 | False | 80 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **737** | Rodney Hood | Utah Jazz | 5.0 | SG | 23.0 | 206.0 | Duke | 1.348440e+06 | False | 80 |
| **738** | Chris Johnson | Utah Jazz | 23.0 | SF | 26.0 | 206.0 | Dayton | 9.813480e+05 | False | 78 |
| **739** | Trey Lyles | Utah Jazz | 41.0 | PF | 20.0 | 234.0 | Kentucky | 2.239800e+06 | False | 82 |

```
df4.drop_duplicates(keep = 'first', inplace = True) # keep may be one of the 'first'/'last'/False
```

```
df4.duplicated().sum()
```

```
0
```

```
df.head()
```

### Encoding the categorical data

```
# Label Encoding of Name, which is not necessary in reality

from sklearn import preprocessing

label_encoder = preprocessing.LabelEncoder()

df4['Name']= label_encoder.fit_transform(df4['Name'])
df4['College']= label_encoder.fit_transform(df4['College'])

df4.sample(10)
```

| | Name | Team | Number | Position | Age | Weight | College | Salary | Salary_Missing | Height_In_Inches |
|---|---|---|---|---|---|---|---|---|---|---|
| **285** | 191 | Miami Heat | 4.0 | PF | 29.0 | 240.0 | 26 | 5.543725e+06 | False | 82 |
| **333** | 265 | Oklahoma City Thunder | 4.0 | PF | 35.0 | 255.0 | 43 | 3.750000e+06 | False | 82 |
| **206** | 66 | Houston Rockets | 33.0 | SG | 30.0 | 186.0 | 28 | 8.229375e+06 | False | 81 |
| **329** | 120 | Minnesota Timberwolves | 4.0 | PF | 25.0 | 250.0 | 30 | 4.630642e+06 | True | 82 |
| **166** | 289 | Indiana Pacers | 25.0 | PF | 24.0 | 250.0 | 89 | 1.007026e+06 | False | 81 |
| **238** | 143 | New Orleans Pelicans | 4.0 | SF | 25.0 | 210.0 | 48 | 8.450590e+05 | False | 79 |
| **359** | 38 | Portland Trail Blazers | 2.0 | PG | 30.0 | 173.0 | 23 | 2.854940e+06 | False | 73 |
| **325** | 369 | Minnesota Timberwolves | 8.0 | PG | 21.0 | 189.0 | 96 | 2.148360e+06 | False | 77 |
| **311** | 112 | Washington Wizards | 17.0 | SG | 30.0 | 195.0 | 46 | 1.100602e+06 | False | 78 |
| **121** | 307 | Sacramento Kings | 8.0 | SF | 29.0 | 230.0 | 20 | 1.240310e+07 | False | 80 |

```
# Ordinal Encoding

from sklearn.preprocessing import OrdinalEncoder
ord1 = OrdinalEncoder()

ord1.fit([df4['Position']])

df4['Position'] = ord1.fit_transform(df4[['Position']])
df4.sample(10)
```

| | Name | Team | Number | Position | Age | Weight | College | Salary | Salary_Missing | Height_In_Inches |
|---|---|---|---|---|---|---|---|---|---|---|
| **303** | 150 | Washington Wizards | 1.0 | 3.0 | 30.0 | 225.0 | 7 | 4.375000e+06 | False | 79 |
| **44** | 204 | Philadelphia 76ers | 5.0 | 2.0 | 24.0 | 200.0 | 66 | 2.144772e+06 | False | 76 |
| **37** | 104 | Philadelphia 76ers | 42.0 | 1.0 | 37.0 | 254.0 | 26 | 4.630642e+06 | True | 81 |
| **356** | 43 | Portland Trail Blazers | 3.0 | 4.0 | 24.0 | 200.0 | 47 | 2.525160e+06 | False | 76 |
| **158** | 340 | Detroit Pistons | 34.0 | 3.0 | 23.0 | 235.0 | 91 | 1.600000e+07 | False | 81 |
| **23** | 124 | Brooklyn Nets | 14.0 | 0.0 | 26.0 | 248.0 | 31 | 9.472760e+05 | False | 82 |
| **252** | 210 | San Antonio Spurs | 23.0 | 4.0 | 33.0 | 199.0 | 111 | 2.006000e+05 | False | 79 |
| **68** | 14 | Golden State Warriors | 9.0 | 3.0 | 32.0 | 215.0 | 1 | 1.171046e+07 | False | 78 |
| **308** | 202 | Washington Wizards | 12.0 | 3.0 | 20.0 | 205.0 | 43 | 1.920240e+06 | False | 79 |
| **251** | 200 | San Antonio Spurs | 2.0 | 3.0 | 24.0 | 230.0 | 85 | 1.640750e+07 | False | 79 |

```
# One_hot Encoding
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()

enc = enc.fit_transform(df4[['Team']]).toarray()
encoded_colm = pd.DataFrame(enc)

df4 = pd.concat([df4, encoded_colm], axis = 1)

df4 = df4.drop(['Team'], axis = 1)
df4.head(10)
```

| | Name | Number | Position | Age | Weight | College | Salary | Salary_Missing | Height_In_Inches | 0 | ... | 20 | 21 | 22 | 23 | 24 | 25 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 28 | 0.0 | 2.0 | 25.0 | 180.0 | 93 | 7.730337e+06 | False | 74 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **1** | 138 | 99.0 | 3.0 | 25.0 | 235.0 | 52 | 6.796117e+06 | False | 78 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **2** | 176 | 30.0 | 4.0 | 27.0 | 205.0 | 8 | 4.630642e+06 | True | 77 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **3** | 287 | 28.0 | 4.0 | 22.0 | 185.0 | 33 | 1.148640e+06 | False | 77 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **4** | 188 | 55.0 | 1.0 | 21.0 | 235.0 | 46 | 1.170960e+06 | False | 80 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **5** | 201 | 41.0 | 0.0 | 25.0 | 238.0 | 35 | 2.165160e+06 | False | 84 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **6** | 335 | 12.0 | 2.0 | 22.0 | 190.0 | 51 | 1.824360e+06 | False | 74 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **7** | 237 | 36.0 | 2.0 | 22.0 | 220.0 | 72 | 3.431040e+06 | False | 76 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **8** | 151 | 7.0 | 0.0 | 24.0 | 260.0 | 70 | 2.569260e+06 | False | 81 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **9** | 129 | 4.0 | 2.0 | 27.0 | 185.0 | 107 | 6.912869e+06 | False | 69 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

10 rows × 39 columns

```
df4.drop(['Salary_Missing'], axis=1, inplace=True)
```

```
df4.head()
```

| | Name | Number | Position | Age | Weight | College | Salary | Height_In_Inches | 0 | 1 | ... | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 28 | 0.0 | 2.0 | 25.0 | 180.0 | 93 | 7.730337e+06 | 74 | 0.0 | 1.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **1** | 138 | 99.0 | 3.0 | 25.0 | 235.0 | 52 | 6.796117e+06 | 78 | 0.0 | 1.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **2** | 176 | 30.0 | 4.0 | 27.0 | 205.0 | 8 | 4.630642e+06 | 77 | 0.0 | 1.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **3** | 287 | 28.0 | 4.0 | 22.0 | 185.0 | 33 | 1.148640e+06 | 77 | 0.0 | 1.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **4** | 188 | 55.0 | 1.0 | 21.0 | 235.0 | 46 | 1.170960e+06 | 80 | 0.0 | 1.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

5 rows × 38 columns

```
df5 = df4.pop('Salary')
df4['Salary']=df5
df4.sample(10)
```

### Feature Scaling - MinMaxScaler

```
from sklearn.preprocessing import MinMaxScaler
array = df4.values
x=array[:,0:37]
y=array[:,37]
scaler=MinMaxScaler(feature_range=(0,1))
```

```
rescaledX=scaler.fit_transform(x)
np.set_printoptions(precision=3)
rescaledX[0:5,:]
```

```
df4.shape
```

### Feature Scaling - StandardScaler

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler().fit(x)
rescaledX=scaler.transform(x)
rescaledX[0:5,:]
```

### Feature Scaling - Normalizer

```
from sklearn.preprocessing import Normalizer
scaler=Normalizer().fit(x)
normalizedX=scaler.transform(x)
normalizedX[0:5,:]
```

### Binarizing the data

```
# Binarizing Data using Binarizer
from sklearn.preprocessing import Binarizer
binarizer=Binarizer(threshold=0.0).fit(x)
binaryX=binarizer.transform(x)
binaryX[0:5,:]
```

### Splitting the Data to Training and Test Sets

```
# split a dataset into train and test sets
from sklearn.model_selection import train_test_split
```

```
# split into train test sets
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.30)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(259, 37) (112, 37) (259,) (112,)
```

```
X_train
```

```
array([[  2.,   9.,   4., ...,   0.,   0.,   0.],
       [197.,  20.,   3., ...,   0.,   0.,   0.],
       [309.,  33.,   1., ...,   0.,   0.,   0.],
       ...,
       [128.,   0.,   2., ...,   0.,   0.,   0.],
       [247.,   4.,   3., ...,   0.,   0.,   0.],
       [ 87.,   1.,   2., ...,   0.,   0.,   0.]])
```