Data Preprocessing

Importing Librbarie

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import QuantileTransformer
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
```

Reading Data

In [ ]:
```
df = pd.read_csv('drawndata1.csv')
df.sample(5)
```

Out[ ]:

	x	y	z
39	483.097265	166.477850	a
214	2666.493774	257.929608	b
211	2542.208129	224.742764	b
196	2137.851893	314.982075	b
170	1627.878054	192.024249	b

Putting first two columns in Data as indepdent variables Just like in house example the size of the house was independent variable

Putting last column as dependent vaiables just like the price of the house was dependent variable

simply I am putting first two columns in X_in, and last columns in y_in

In [ ]:
```
X_in=df[['x','y']]
y_in=df['z']# this y_in contains both 'a' and 'b'
```

But in the code below I am putting only 'a' in "y_with_only_a" to plot in different colors

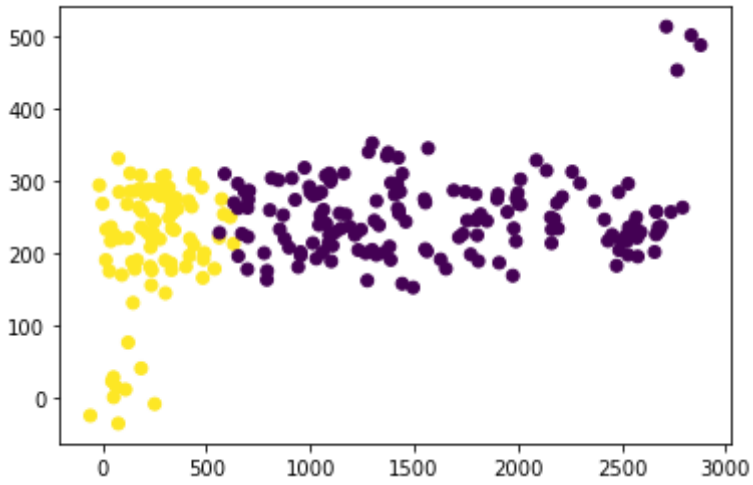
then in second line I am putting only 'b' in "y_with_only_b" to plot in different colors

In [ ]:
```
y_with_only_a = df['z']=='a'# this contains only 'a'
y_with_only_b = df['z']=='b'# this contains only 'b'
```

The plot below is scatter on plot on raw data with colors on 'a'

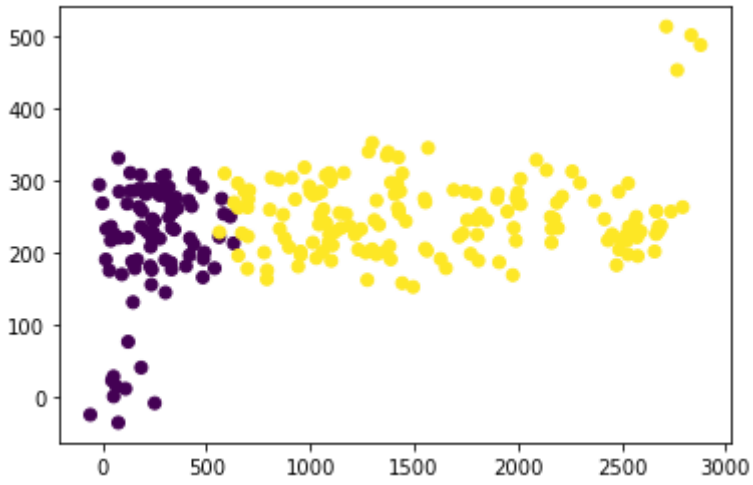
In [ ]:
```
#on pure raw data
plt.scatter(X_in['x'],X_in['y'],c=y_with_only_a)
                            # the c is used for color, c=y_with_only_a it
                            # means that color the data that is inside of variable
                            # and we know that this varaible contains only 'a' poi
                            # you can see results in graph that 'a' is colored yel
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x1dc16a53ca0>
```



In [ ]:
```
#on pure raw data
plt.scatter(X_in['x'],X_in['y'],c=y_with_only_b)
                            # the c is used for color, c=y_with_only_b it
                            # means that color the data that is inside of variable
                            # and we know that this varaible contains only 'b' poi
                            # you can see results in graph that 'b' is colored yel
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x1dc16ab5220>
```



In the above scatter plot you can observe outliers

Now we will use two data preprocessing techniques to and check which one is working good:

1. StandardScaler()

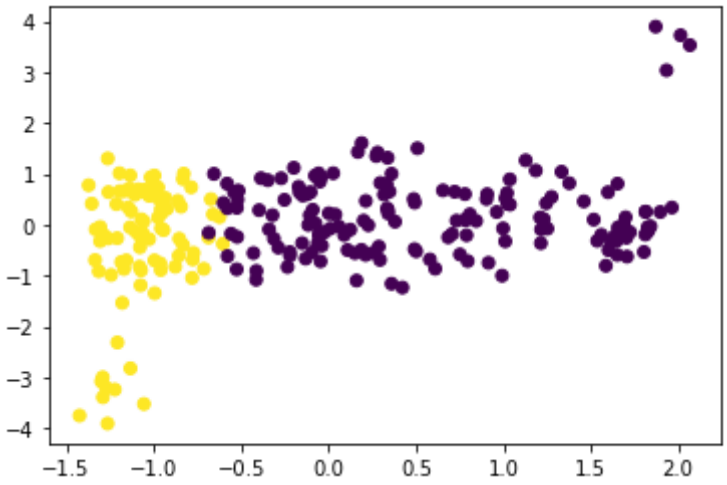
2. QuantileTransformer()

In [ ]:
```
scaled_X = StandardScaler().fit_transform(X_in)
```

Drawing scatter plot on transformed data

In [ ]:
```
plt.scatter(scaled_X[:,0],scaled_X[:,1],c=y_with_only_a)
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x1dc16b10940>
```



we can see from above graph that outliers still have great effect so thie method is useless

Lets try Quantile transformer

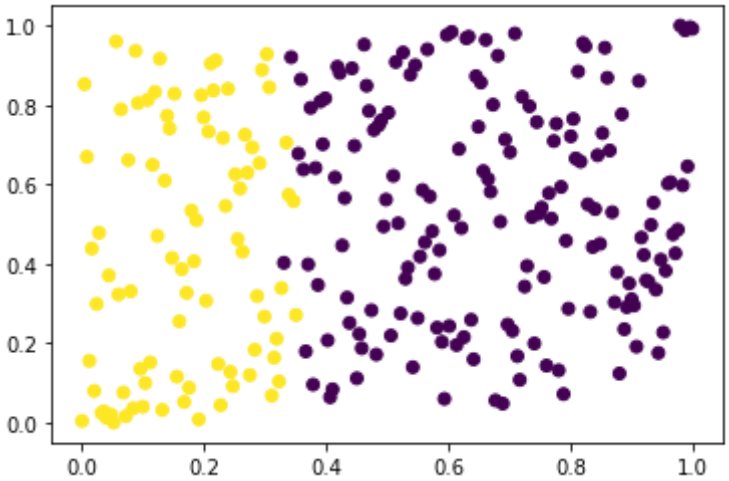
In [ ]:
```
scaled_X1 = QuantileTransformer().fit_transform(X_in)
```

```
c:\Users\AbdulRaheemShahzad\anaconda3\lib\site-packages\sklearn\preprocessing\_data.py:2590: UserWarning: n_quantiles (1000) is greater than the total number of samples (252). n_quantiles is set to n_samples.
  warnings.warn(
```

plot after Quantile transformer

In [ ]:
```
plt.scatter(scaled_X1[:,0],scaled_X1[:,1],c=y_with_only_a)
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x1dc16b827f0>
```



Now you can see Quantile transformers is working perfectly fine becuase it is made to handle outilers

Keypoint:

- there are differnt methods for preprocessing the data you need to select pre-processing method acording to your data.