Label Encoder

In [1]:

```
from sklearn.preprocessing import LabelEncoder
import pandas as pd
```

In [2]:

```
data =pd.read_csv("C:/Users/Ashraf/Documents/Datafiles/iris.csv",index_col=0)
```

In [3]:

```
data.head()
```

Out[3]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa

In [4]:

```
#Changes does not effect the "data" dataframe
data1=data.copy()
```

In [5]:

```
labelencoder = LabelEncoder()
data1.iloc[:, -1] = labelencoder.fit_transform(data1.iloc[:,-1])
```

In [6]:

```
data1
```

Out[6]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	0
2	4.9	3.0	1.4	0.2	0
3	4.7	3.2	1.3	0.2	0
4	4.6	3.1	1.5	0.2	0
5	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
146	6.7	3.0	5.2	2.3	2
147	6.3	2.5	5.0	1.9	2
148	6.5	3.0	5.2	2.0	2
149	6.2	3.4	5.4	2.3	2
150	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

One Hot Encoder

Using sklearn

In [7]:

```
from sklearn.preprocessing import OneHotEncoder
```

In [8]:

```
data2=pd.read_csv("C:/Users/Ashraf/Documents/Datafiles/iris.csv",index_col=0)
```

In [9]:

```
data2
```

Out[9]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
146	6.7	3.0	5.2	2.3	virginica
147	6.3	2.5	5.0	1.9	virginica
148	6.5	3.0	5.2	2.0	virginica
149	6.2	3.4	5.4	2.3	virginica
150	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [10]:

```
# creating instance of one-hot-encoder
enc = OneHotEncoder(handle_unknown='ignore')
```

In [11]:

```
# passing bridge-types-cat column (label encoded values of bridge_types)
enc_df = pd.DataFrame(enc.fit_transform(data2[['Species']]).toarray())
```

In [12]:

```
enc_df
```

Out[12]:

	0	1	2
0	1.0	0.0	0.0
1	1.0	0.0	0.0
2	1.0	0.0	0.0
3	1.0	0.0	0.0
4	1.0	0.0	0.0
...	...	...	...
145	0.0	0.0	1.0
146	0.0	0.0	1.0
147	0.0	0.0	1.0
148	0.0	0.0	1.0
149	0.0	0.0	1.0

150 rows × 3 columns

In [13]:

```
# merge with main df
data_final = data2.iloc[:,0:4].join(enc_df)
data_final
```

Out[13]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	0	1	2
1	5.1	3.5	1.4	0.2	1.0	0.0	0.0
2	4.9	3.0	1.4	0.2	1.0	0.0	0.0
3	4.7	3.2	1.3	0.2	1.0	0.0	0.0
4	4.6	3.1	1.5	0.2	1.0	0.0	0.0
5	5.0	3.6	1.4	0.2	1.0	0.0	0.0
...	...	...	...	...	...	...	...
146	6.7	3.0	5.2	2.3	0.0	0.0	1.0
147	6.3	2.5	5.0	1.9	0.0	0.0	1.0
148	6.5	3.0	5.2	2.0	0.0	0.0	1.0
149	6.2	3.4	5.4	2.3	0.0	0.0	1.0
150	5.9	3.0	5.1	1.8	NaN	NaN	NaN

150 rows × 7 columns

Using Pandas

In [14]:

```
import pandas as pd
```

In [15]:

```
data3 =pd.read_csv("C:/Users/Ashraf/Documents/Datafiles/iris.csv",index_col=0)
```

In [16]:

```
data_encoded=pd.get_dummies(data3)
```

In [17]:

```
data_encoded
```

Out[17]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species_setosa	Species_versicolor	Species_virgini
1	5.1	3.5	1.4	0.2	1	0	
2	4.9	3.0	1.4	0.2	1	0	
3	4.7	3.2	1.3	0.2	1	0	
4	4.6	3.1	1.5	0.2	1	0	
5	5.0	3.6	1.4	0.2	1	0	
...	...	...	...	...	...	...	
146	6.7	3.0	5.2	2.3	0	0	
147	6.3	2.5	5.0	1.9	0	0	
148	6.5	3.0	5.2	2.0	0	0	
149	6.2	3.4	5.4	2.3	0	0	

IsolationForest

In [18]:

```
from sklearn.ensemble import IsolationForest
```

In [19]:

```
data =pd.read_csv("C:/Users/Ashraf/Documents/Datafiles/iris.csv",index_col=0)
data_encoded=pd.get_dummies(data)
```

In [20]:

```
# training the model
clf = IsolationForest(random_state=10,contamination=.01)
clf.fit(data_encoded)
```

Out[20]:

```
IsolationForest(contamination=0.01, random_state=10)
```

In [21]:

```
# predictions
y_pred_outliers = clf.predict(data_encoded)
```

In [22]:

```
#-1 for outliers and 1 for inliers.
y_pred_outliers
```

Out[22]:

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

In [23]:

```
data_encoded
```

Out[23]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species_setosa	Species_versicolor
1	5.1	3.5	1.4	0.2	1	0
2	4.9	3.0	1.4	0.2	1	0
3	4.7	3.2	1.3	0.2	1	0
4	4.6	3.1	1.5	0.2	1	0
5	5.0	3.6	1.4	0.2	1	0
...	...	...	...	...	...	...
146	6.7	3.0	5.2	2.3	0	0
147	6.3	2.5	5.0	1.9	0	0
148	6.5	3.0	5.2	2.0	0	0
149	6.2	3.4	5.4	2.3	0	0
150	5.9	3.0	5.1	1.8	0	0

150 rows × 7 columns

In [24]:

```
## Let us add a new data point which is outlier
data_encoded.loc[150]=[20,40,30,50,1,0,0]
```

In [25]:

```
data_encoded
```

Out[25]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species_setosa	Species_versicolor
1	5.1	3.5	1.4	0.2	1	0
2	4.9	3.0	1.4	0.2	1	0
3	4.7	3.2	1.3	0.2	1	0
4	4.6	3.1	1.5	0.2	1	0
5	5.0	3.6	1.4	0.2	1	0
...	...	...	...	...	...	...
146	6.7	3.0	5.2	2.3	0	0
147	6.3	2.5	5.0	1.9	0	0
148	6.5	3.0	5.2	2.0	0	0
149	6.2	3.4	5.4	2.3	0	0
150	20.0	40.0	30.0	50.0	1	0

150 rows × 7 columns

In [26]:

```
# training the model
clf = IsolationForest(random_state=10,contamination=.01)
clf.fit(data_encoded)
# predictions
y_pred_outliers = clf.predict(data_encoded)
y_pred_outliers
```

Out[26]:

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

In [27]:

```
data_encoded['scores']=clf.decision_function(data_encoded)
```

In [28]:

```
data_encoded['anomaly']=clf.predict(data_encoded.iloc[:,0:7])
```

In [29]:

```
data_encoded
```

Out[29]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species_setosa	Species_versicolor
1	5.1	3.5	1.4	0.2	1	0
2	4.9	3.0	1.4	0.2	1	0
3	4.7	3.2	1.3	0.2	1	0
4	4.6	3.1	1.5	0.2	1	0
5	5.0	3.6	1.4	0.2	1	0
...	...	...	...	...	...	...
146	6.7	3.0	5.2	2.3	0	0
147	6.3	2.5	5.0	1.9	0	0
148	6.5	3.0	5.2	2.0	0	0
149	6.2	3.4	5.4	2.3	0	0
150	20.0	40.0	30.0	50.0	1	0

150 rows × 9 columns

In [30]:

```
#Print the outlier data points
data_encoded[data_encoded['anomaly']==-1]
```

Out[30]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species_setosa	Species_versicolor
107	4.9	2.5	4.5	1.7	0	0
150	20.0	40.0	30.0	50.0	1	0

PPS score

In [31]:

```
#install the package
!pip install ppscore
```

```
Requirement already satisfied: ppscore in c:\users\ashraf\anaconda3\lib\site
-packages (1.2.0)
Requirement already satisfied: scikit-learn<1.0.0,>=0.20.2 in c:\users\ashra
f\anaconda3\lib\site-packages (from ppscore) (0.24.2)
Requirement already satisfied: pandas<2.0.0,>=1.0.0 in c:\users\ashraf\anaco
nda3\lib\site-packages (from ppscore) (1.3.4)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\ashraf\ana
conda3\lib\site-packages (from pandas<2.0.0,>=1.0.0->ppscore) (2.8.2)
Requirement already satisfied: numpy>=1.17.3 in c:\users\ashraf\anaconda3\li
b\site-packages (from pandas<2.0.0,>=1.0.0->ppscore) (1.20.3)
Requirement already satisfied: pytz>=2017.3 in c:\users\ashraf\anaconda3\lib
\site-packages (from pandas<2.0.0,>=1.0.0->ppscore) (2021.3)
Requirement already satisfied: six>=1.5 in c:\users\ashraf\anaconda3\lib\sit
e-packages (from python-dateutil>=2.7.3->pandas<2.0.0,>=1.0.0->ppscore) (1.1
6.0)
Requirement already satisfied: joblib>=0.11 in c:\users\ashraf\anaconda3\lib
\site-packages (from scikit-learn<1.0.0,>=0.20.2->ppscore) (1.0.1)
Requirement already satisfied: scipy>=0.19.1 in c:\users\ashraf\anaconda3\li
b\site-packages (from scikit-learn<1.0.0,>=0.20.2->ppscore) (1.7.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\ashraf\anaco
nda3\lib\site-packages (from scikit-learn<1.0.0,>=0.20.2->ppscore) (2.2.0)
```

In [32]:

```
import ppscore as pps
```

In [33]:

```
#pps.score(df, "feature_column", "target_column")  syntax
pps.score(data, "Sepal.Length", "Petal.Length")
```

Out[33]:

```
{'x': 'Sepal.Length',
 'y': 'Petal.Length',
 'ppscore': 0.550422595049248,
 'case': 'regression',
 'is_valid_score': True,
 'metric': 'mean absolute error',
 'baseline_score': 1.4886666666666668,
 'model_score': 0.6692708968366863,
 'model': DecisionTreeRegressor()}
```

In [34]:

```
#calculate the whole PPS matrix
pps.matrix(data)
```

Out[34]:

	x	y	ppscore	case	is_valid_score	metric	baseline_score
0	Sepal.Length	Sepal.Length	1.000000	predict_itself	True	None	0.000000
1	Sepal.Length	Sepal.Width	0.000000	regression	True	mean absolute error	0.330667
2	Sepal.Length	Petal.Length	0.550423	regression	True	mean absolute error	1.488667
3	Sepal.Length	Petal.Width	0.431739	regression	True	mean absolute error	0.644667
4	Sepal.Length	Species	0.471649	classification	True	weighted F1	0.353333
5	Sepal.Width	Sepal.Length	0.006966	regression	True	mean absolute error	0.684667
6	Sepal.Width	Sepal.Width	1.000000	predict_itself	True	None	0.000000
7	Sepal.Width	Petal.Length	0.172375	regression	True	mean absolute error	1.488667
8	Sepal.Width	Petal.Width	0.132858	regression	True	mean absolute error	0.644667
9	Sepal.Width	Species	0.156915	classification	True	weighted F1	0.353333
10	Petal.Length	Sepal.Length	0.525617	regression	True	mean absolute error	0.684667
11	Petal.Length	Sepal.Width	0.052136	regression	True	mean absolute error	0.330667
12	Petal.Length	Petal.Length	1.000000	predict_itself	True	None	0.000000
13	Petal.Length	Petal.Width	0.744945	regression	True	mean absolute error	0.644667
14	Petal.Length	Species	0.884812	classification	True	weighted F1	0.353333
15	Petal.Width	Sepal.Length	0.383911	regression	True	mean absolute error	0.684667
16	Petal.Width	Sepal.Width	0.254616	regression	True	mean absolute error	0.330667
17	Petal.Width	Petal.Length	0.798274	regression	True	mean absolute error	1.488667
18	Petal.Width	Petal.Width	1.000000	predict_itself	True	None	0.000000

	x	y	ppscore	case	is_valid_score	metric	baseline_score
19	Petal.Width	Species	0.927652	classification	True	weighted F1	0.353333
20	Species	Sepal.Length	0.409634	regression	True	mean absolute error	0.684667
21	Species	Sepal.Width	0.194307	regression	True	mean absolute error	0.330667
22	Species	Petal.Length	0.785393	regression	True	mean absolute error	1.488667
23	Species	Petal.Width	0.755749	regression	True	mean absolute error	0.644667
24	Species	Species	1.000000	predict_itself	True	None	0.000000

In [ ]: