```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
%matplotlib inline

file_path ='/content/Classified_Data.txt'
df = pd.read_table(file_path, sep=',', index_col = 0)

df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 1000,\n  \"fields\":
[\n    {\n      \"column\": \"WTT\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.28963525165758874,\n
\"min\": 0.174411668391638,\n        \"max\": 1.721779168965468,\n
\"num_unique_values\": 1000,\n        \"samples\": [\n
0.7063010303254464,\n          1.3496574995895918,\n
1.3169005830819778\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"PTI\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.25708526213795485,\n        \"min\": 0.441398100295989,\n
\"max\": 1.8337565522536252,\n        \"num_unique_values\": 1000,\n
\"samples\": [\n          1.521177618895161,\n
1.2203336025956588,\n          1.1486493499359351\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"EQW\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.2915538503444115,\n
\"min\": 0.1709236280526556,\n        \"max\": 1.7227247553711322,\n
\"num_unique_values\": 1000,\n        \"samples\": [\n
1.130807438901416,\n          0.7058621252266584,\n
1.0433060732882415\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"SBI\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.22964502416144614,\n        \"min\": 0.0450266664094166,\n
\"max\": 1.634884045436437,\n        \"num_unique_values\": 1000,\n
\"samples\": [\n          0.4034386350051173,\n
0.8143574265988366,\n          0.9019193027640824\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"LQE\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.24341295346925404,\n
\"min\": 0.3153070077960995,\n        \"max\": 1.650049589008639,\n
\"num_unique_values\": 1000,\n        \"samples\": [\n
0.8967456416302068,\n          0.6699170570739925,\n
0.9443602753750552\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"QWG\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.25612059661087283,\n        \"min\": 0.2623888468883443,\n
\"max\": 1.6669023520657231,\n        \"num_unique_values\": 1000,\n
```

```
\"samples\": [\n          0.9158571059683956,\n
0.6719997779429592,\n          1.1087710280109884\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"FDJ\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.25511802913126286,\n
\"min\": 0.2952280855806717,\n        \"max\": 1.7133422293242386,\n
\"num_unique_values\": 1000,\n        \"samples\": [\n
1.0701452103715168,\n          0.7467547171292814,\n
0.3132418138897407\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"PJF\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.28898164327767917,\n        \"min\": 0.299475657020008,\n
\"max\": 1.7854196250383634,\n        \"num_unique_values\": 1000,\n
\"samples\": [\n          1.2150324734826512,\n
0.5407172165654959,\n          0.8107025990833856\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"HQE\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.29373751661647246,\n
\"min\": 0.3651566098613977,\n        \"max\": 1.8856900849797629,\n
\"num_unique_values\": 1000,\n        \"samples\": [\n
1.0385048132735202,\n          1.4319928464192602,\n
1.0343550687529062\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"NXJ\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.204225023410037,\n        \"min\": 0.639692747423801,\n
\"max\": 1.8939496030653464,\n        \"num_unique_values\": 1000,\n
\"samples\": [\n          1.4457974108545717,\n
1.173152878750532,\n          1.5851535299635755\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"TARGET CLASS\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```


```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000 entries, 0 to 999
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   WTT           1000 non-null   float64
 1   PTI           1000 non-null   float64
 2   EQW           1000 non-null   float64
 3   SBI           1000 non-null   float64
 4   LQE           1000 non-null   float64
 5   QWG           1000 non-null   float64
```

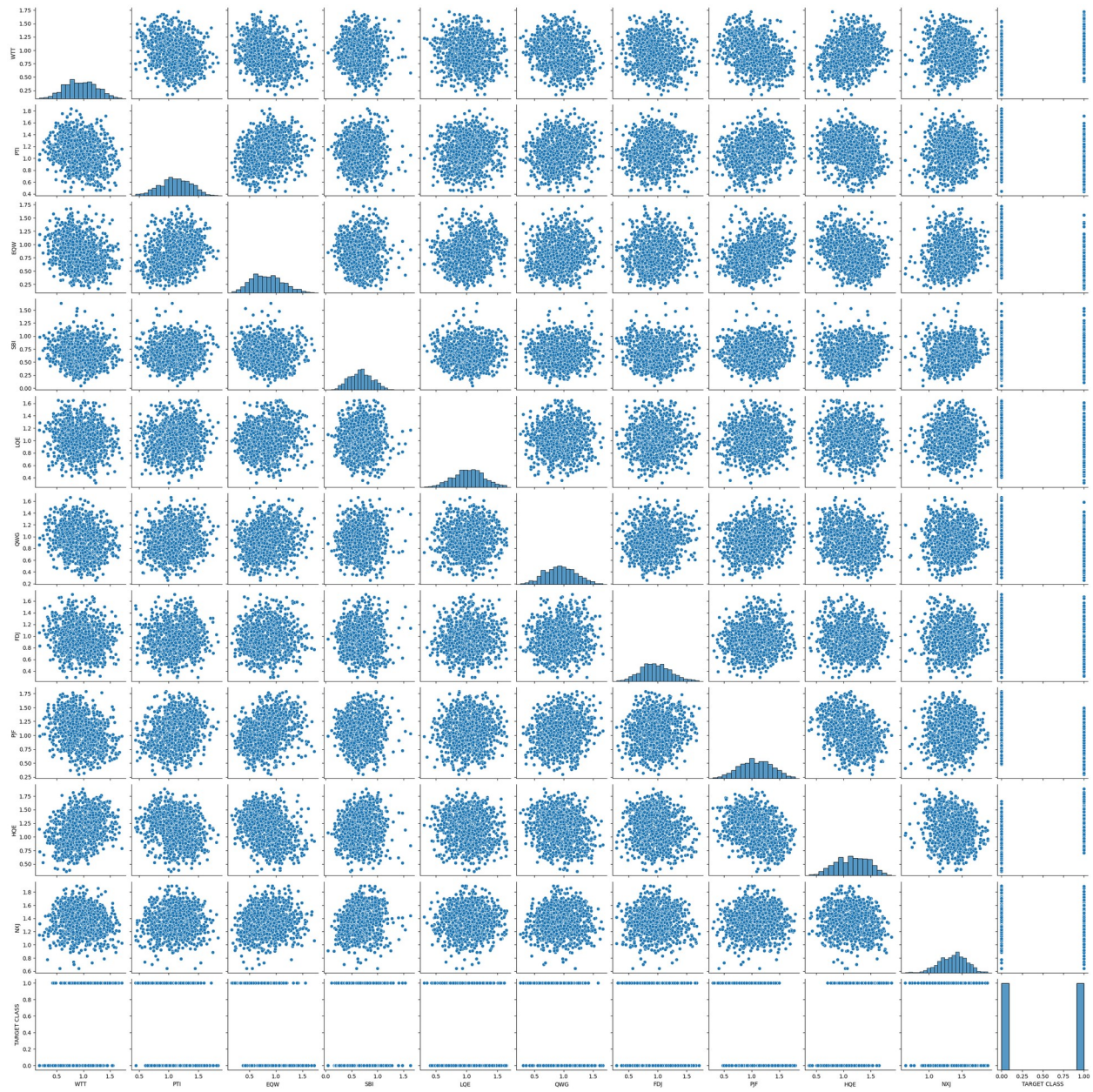
```
 6   FDJ           1000 non-null   float64
 7   PJF           1000 non-null   float64
 8   HQE           1000 non-null   float64
 9   NXJ           1000 non-null   float64
 10  TARGET CLASS  1000 non-null   int64
dtypes: float64(10), int64(1)
memory usage: 93.8 KB

summary = df.describe(percentiles=[0.25, 0.5, 0.75, 0.90])
print(summary)

                WTT          PTI          EQW          SBI          LQE
\
count  1000.000000  1000.000000  1000.000000  1000.000000  1000.000000

mean      0.949682     1.114303     0.834127     0.682099     1.032336

std       0.289635     0.257085     0.291554     0.229645     0.243413

min       0.174412     0.441398     0.170924     0.045027     0.315307

25%       0.742358     0.942071     0.615451     0.515010     0.870855

50%       0.940475     1.118486     0.813264     0.676835     1.035824

75%       1.163295     1.307904     1.028340     0.834317     1.198270

90%       1.336612     1.441901     1.223127     0.983470     1.341138

max       1.721779     1.833757     1.722725     1.634884     1.650050


                QWG          FDJ          PJF          HQE          NXJ
\
count  1000.000000  1000.000000  1000.000000  1000.000000  1000.000000

mean      0.943534     0.963422     1.071960     1.158251     1.362725

std       0.256121     0.255118     0.288982     0.293738     0.204225

min       0.262389     0.295228     0.299476     0.365157     0.639693

25%       0.761064     0.784407     0.866306     0.934340     1.222623

50%       0.941502     0.945333     1.065500     1.165556     1.375368

75%       1.123060     1.134852     1.283156     1.383173     1.504832

90%       1.277552     1.306497     1.452713     1.535520     1.618096

max       1.666902     1.713342     1.785420     1.885690     1.893950
```

```
       TARGET CLASS
count    1000.00000
mean        0.50000
std         0.50025
min         0.00000
25%         0.00000
50%         0.50000
75%         1.00000
90%         1.00000
max         1.00000
```

```
df.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000 entries, 0 to 999
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   WTT           1000 non-null   float64
 1   PTI           1000 non-null   float64
 2   EQW           1000 non-null   float64
 3   SBI           1000 non-null   float64
 4   LQE           1000 non-null   float64
 5   QWG           1000 non-null   float64
 6   FDJ           1000 non-null   float64
 7   PJF           1000 non-null   float64
 8   HQE           1000 non-null   float64
 9   NXJ           1000 non-null   float64
 10  TARGET CLASS  1000 non-null   int64  
dtypes: float64(10), int64(1)
memory usage: 93.8 KB
```

```
sns.pairplot(df)
plt.show()
```

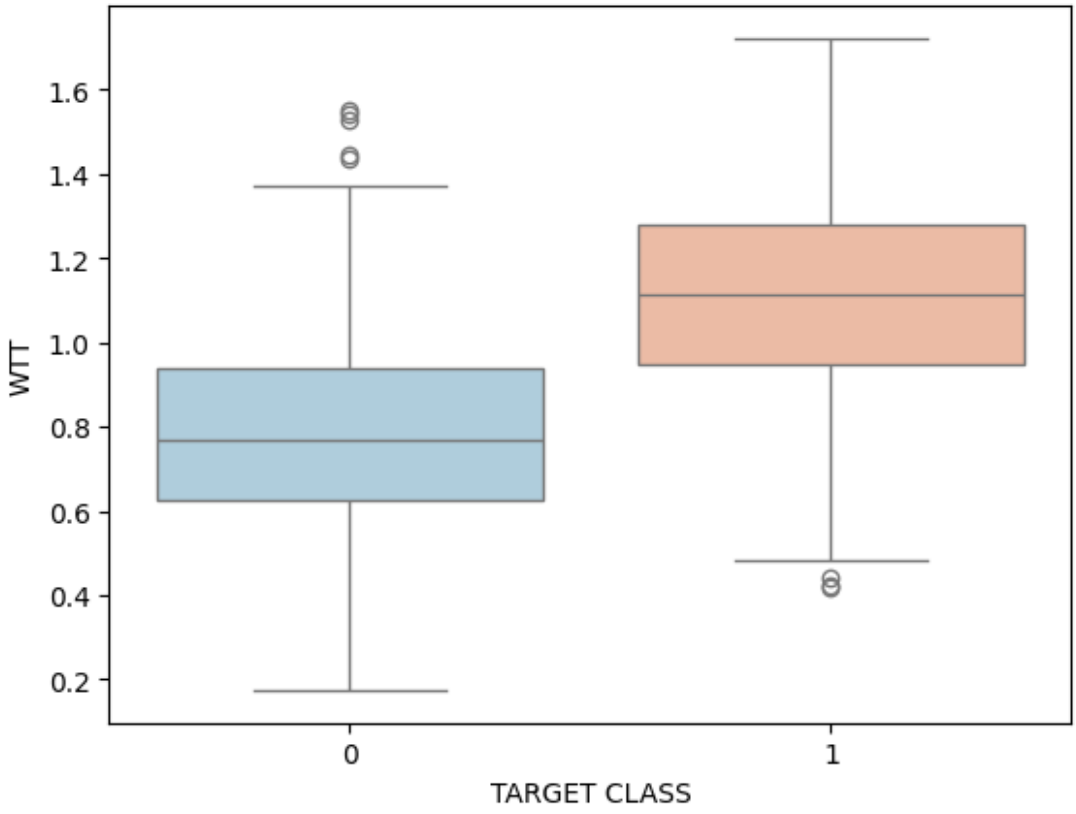


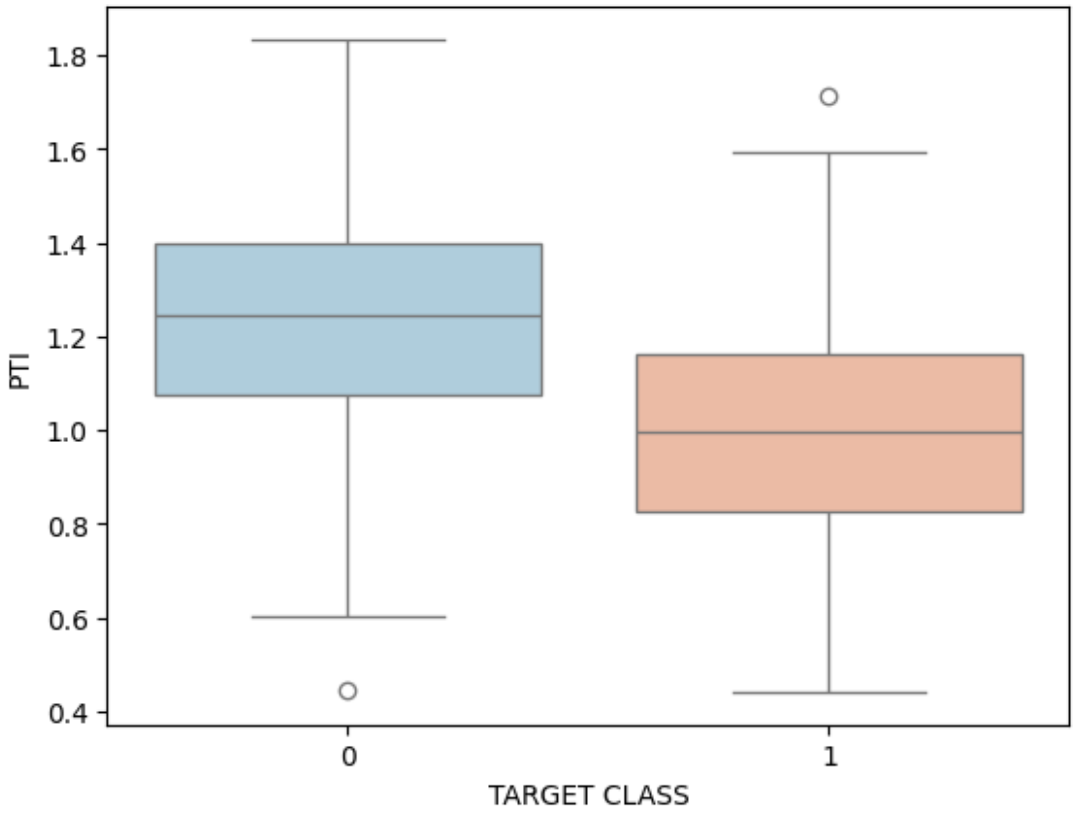

```
l = list(df.columns)
l[0:len(l)-2]
for i in range(len(l)-1):
  sns.boxplot(x='TARGET CLASS', y=l[i], data=df, palette='RdBu_r')
  plt.figure()
```

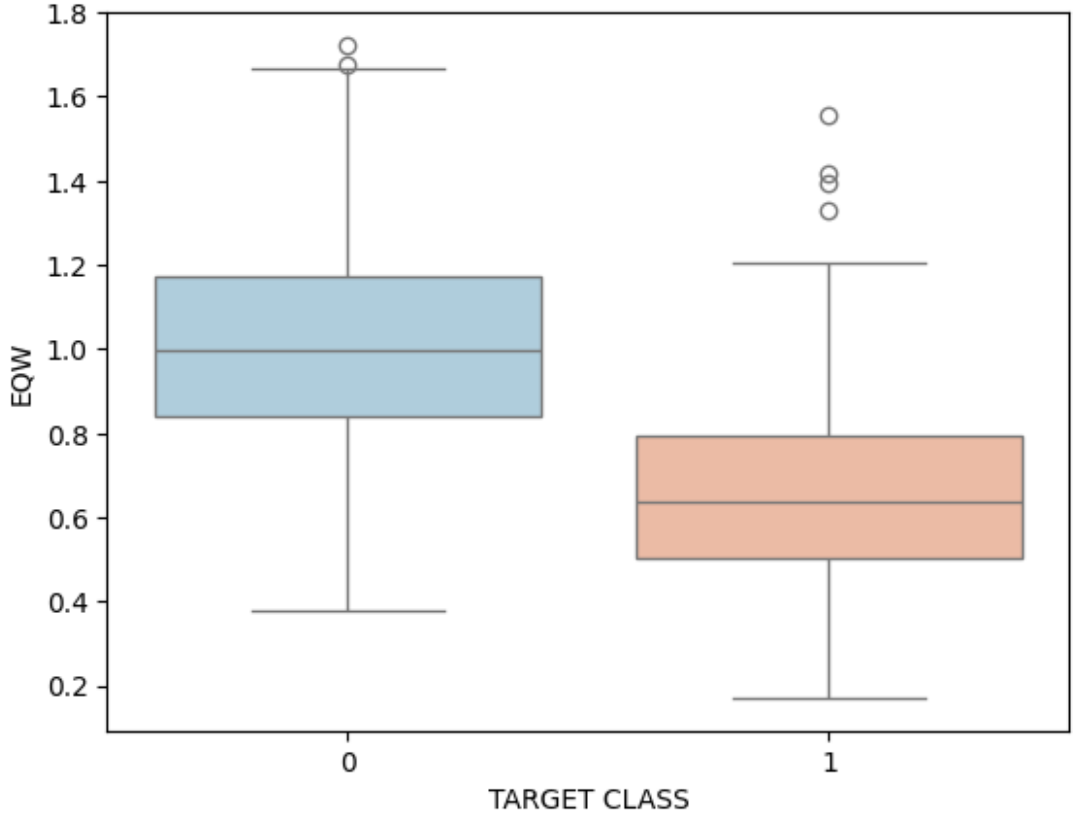
```
<ipython-input-16-4b7f2990ed58>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.
```

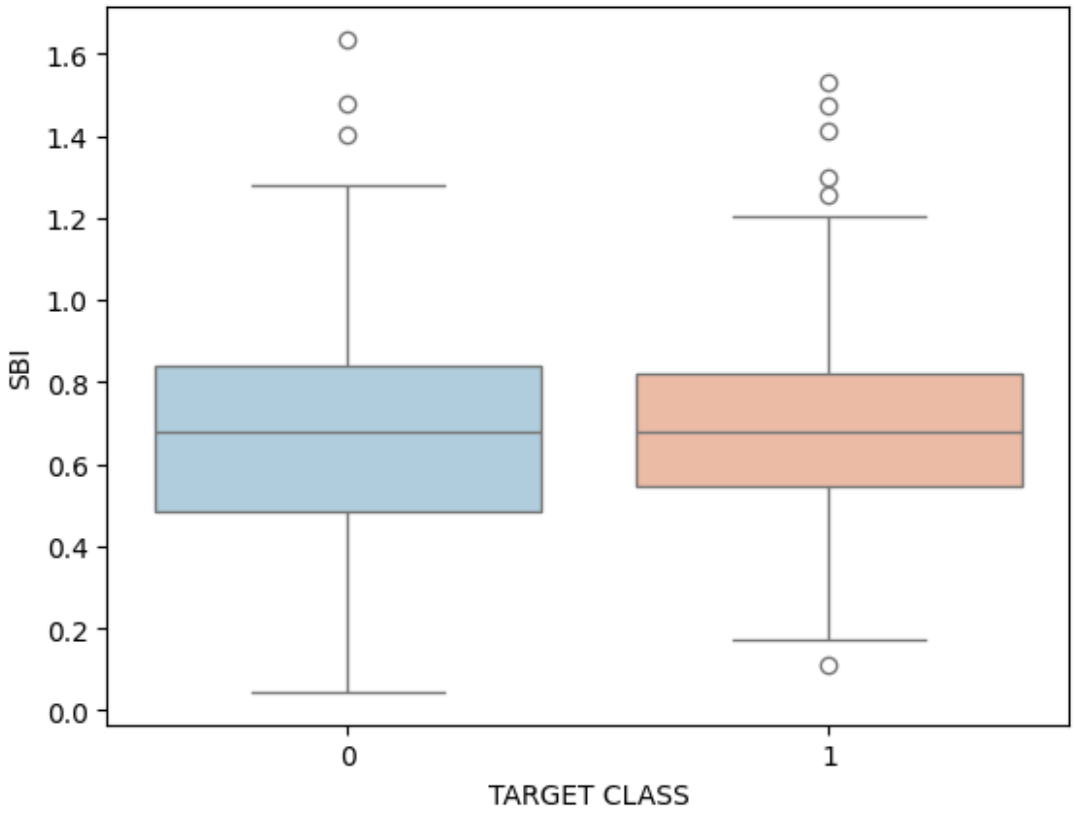
```
  sns.boxplot(x='TARGET CLASS', y=l[i], data=df, palette='RdBu_r')
<ipython-input-16-4b7f2990ed58>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(x='TARGET CLASS', y=l[i], data=df, palette='RdBu_r')
<ipython-input-16-4b7f2990ed58>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(x='TARGET CLASS', y=l[i], data=df, palette='RdBu_r')
<ipython-input-16-4b7f2990ed58>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(x='TARGET CLASS', y=l[i], data=df, palette='RdBu_r')
<ipython-input-16-4b7f2990ed58>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(x='TARGET CLASS', y=l[i], data=df, palette='RdBu_r')
<ipython-input-16-4b7f2990ed58>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(x='TARGET CLASS', y=l[i], data=df, palette='RdBu_r')
<ipython-input-16-4b7f2990ed58>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(x='TARGET CLASS', y=l[i], data=df, palette='RdBu_r')
<ipython-input-16-4b7f2990ed58>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(x='TARGET CLASS', y=l[i], data=df, palette='RdBu_r')
<ipython-input-16-4b7f2990ed58>:4: FutureWarning:
```

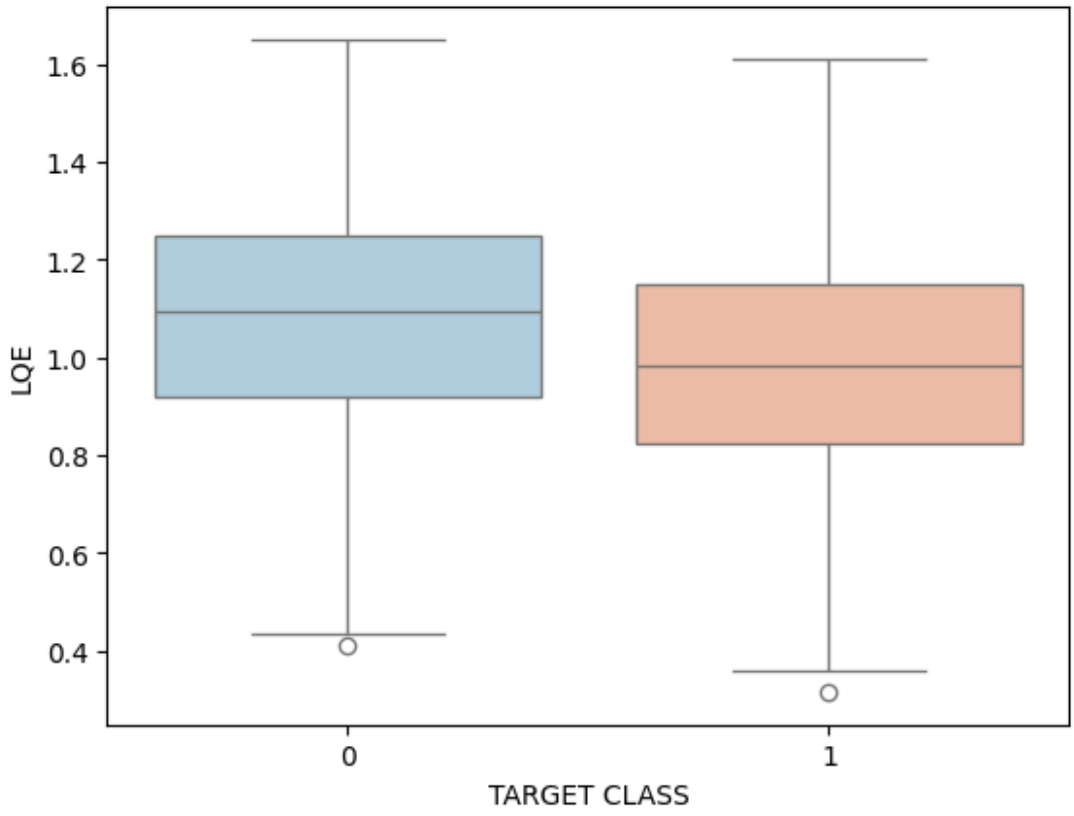
```
Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(x='TARGET CLASS', y=l[i], data=df, palette='RdBu_r')
<ipython-input-16-4b7f2990ed58>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(x='TARGET CLASS', y=l[i], data=df, palette='RdBu_r')
```

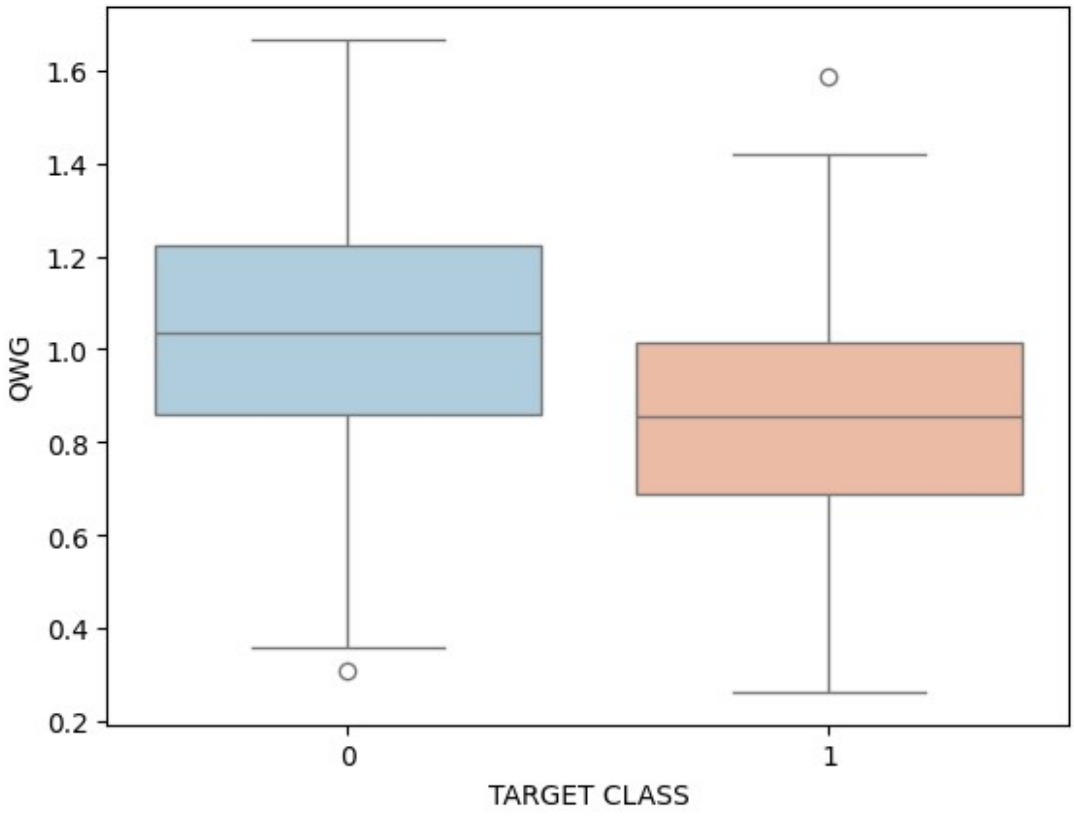


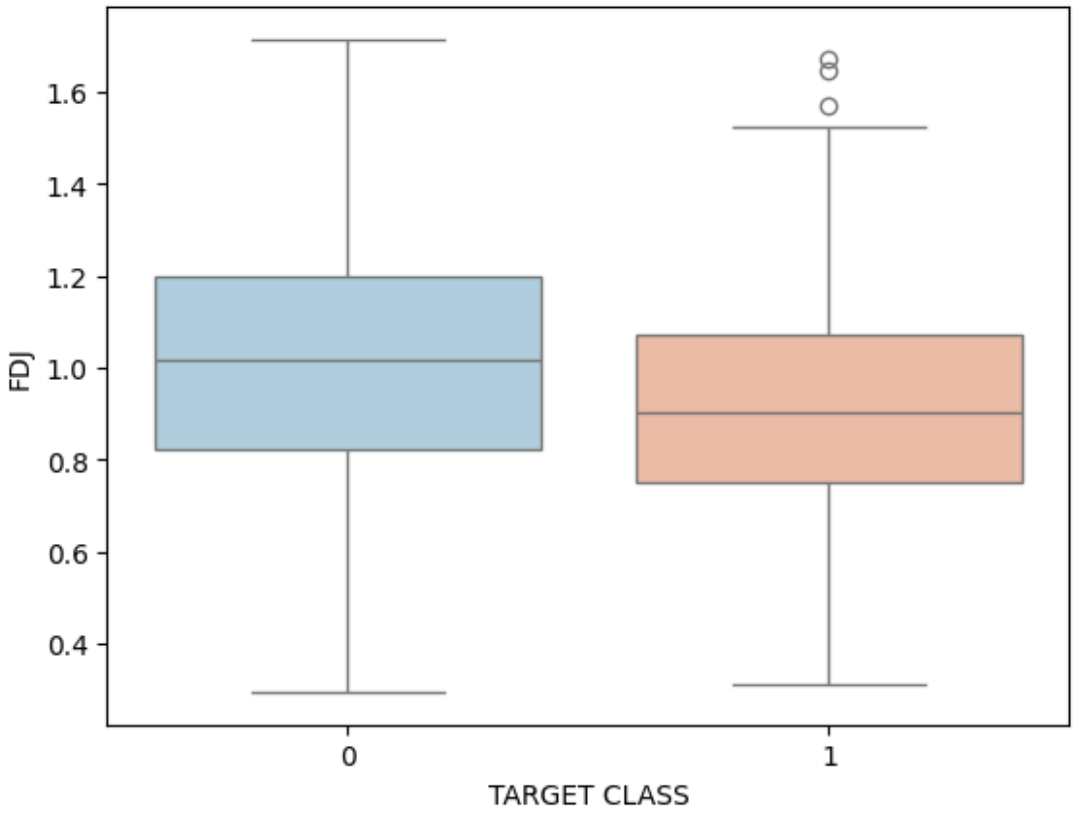

PTI
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
1
TARGET CLASS

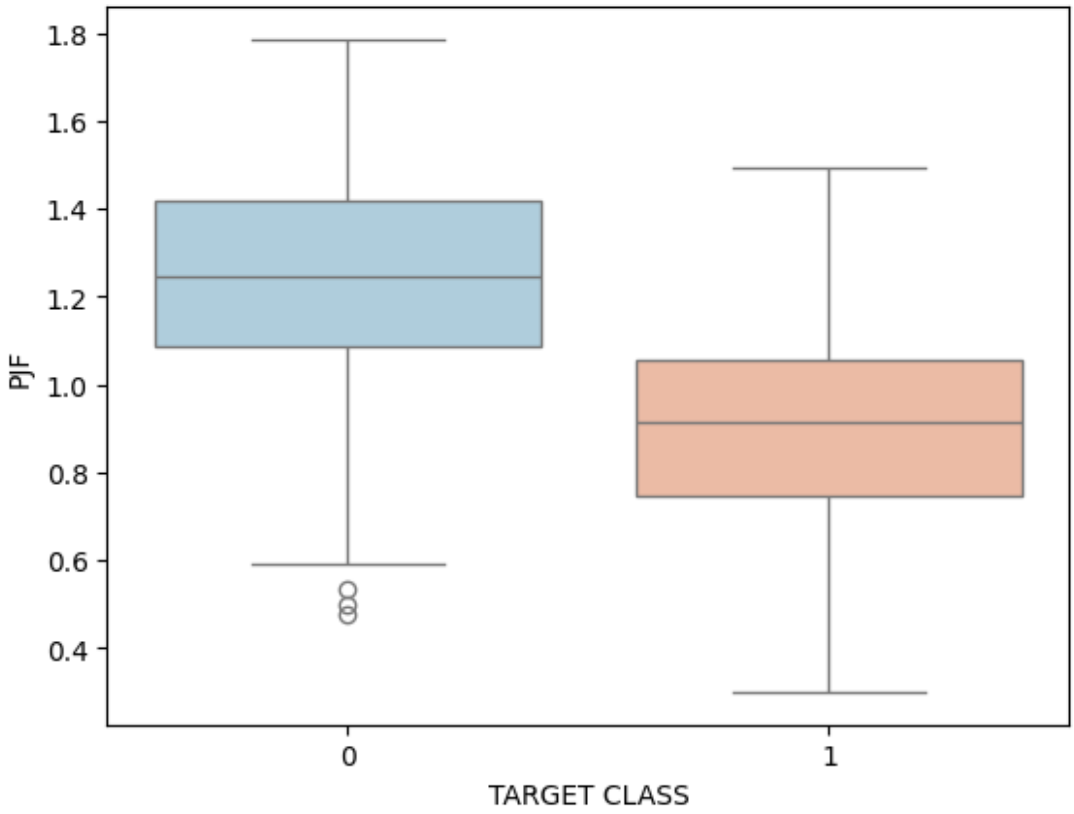

EQW
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0
1
TARGET CLASS

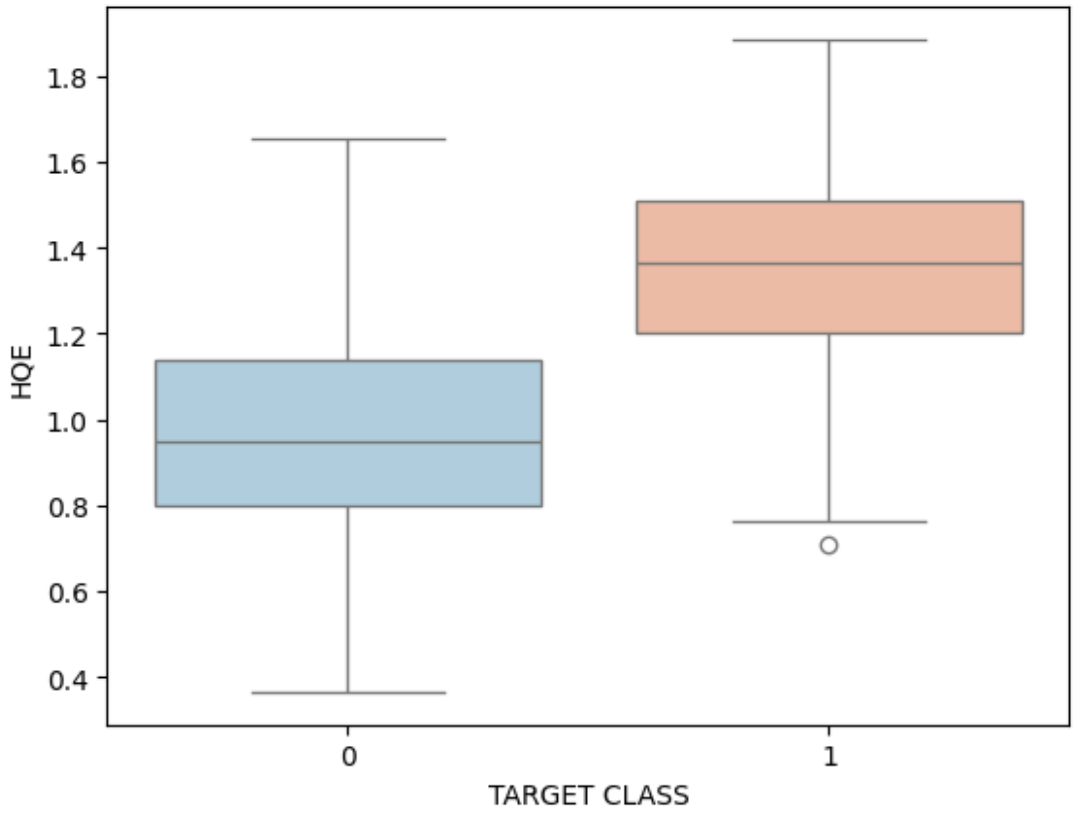

SBI
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
1
TARGET CLASS

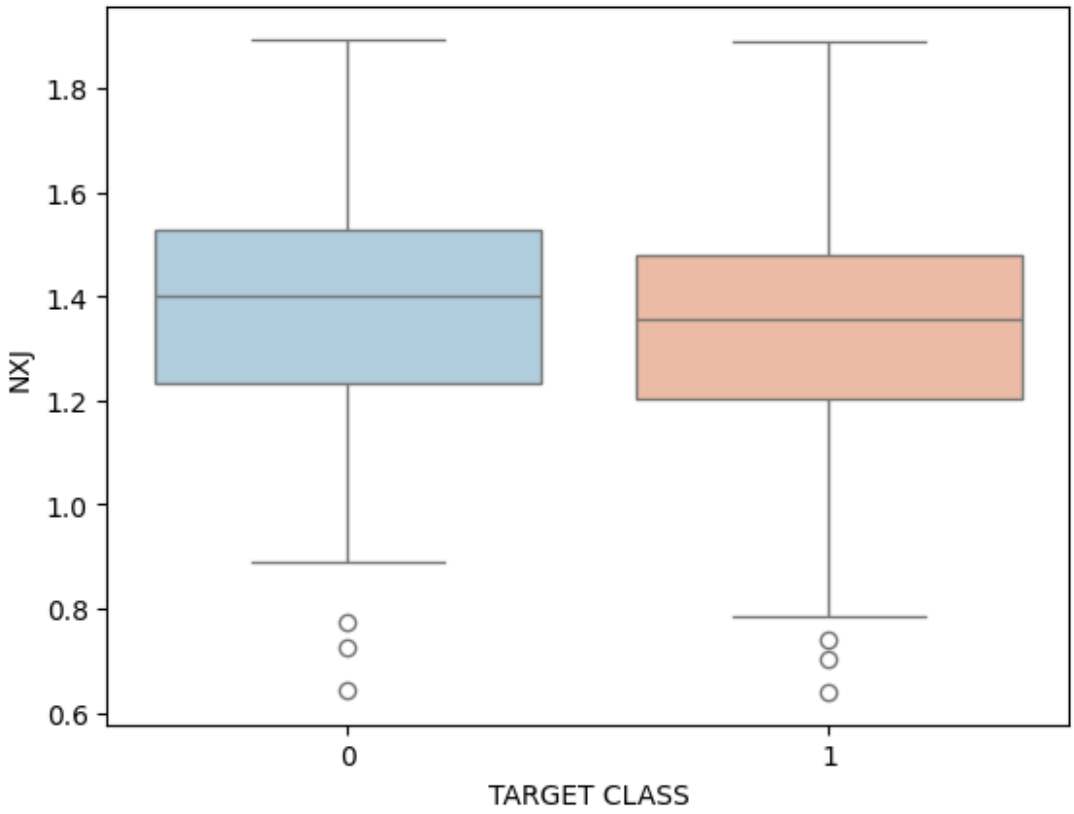

LQE
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
1
TARGET CLASS


QWG
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0
1
TARGET CLASS


FDJ
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
1
TARGET CLASS


PJF
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
1
TARGET CLASS


HQE
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
1
TARGET CLASS




```
<Figure size 640x480 with 0 Axes>
```

```
from sklearn.preprocessing import StandardScaler
Scaler = StandardScaler()

Scaler.fit(df.drop('TARGET CLASS',axis=1))
scaled_features = Scaler.transform(df.drop('TARGET CLASS',axis=1))

df_feat = pd.DataFrame(scaled_features, columns = df.columns[:-1])
df_feat.head()
```

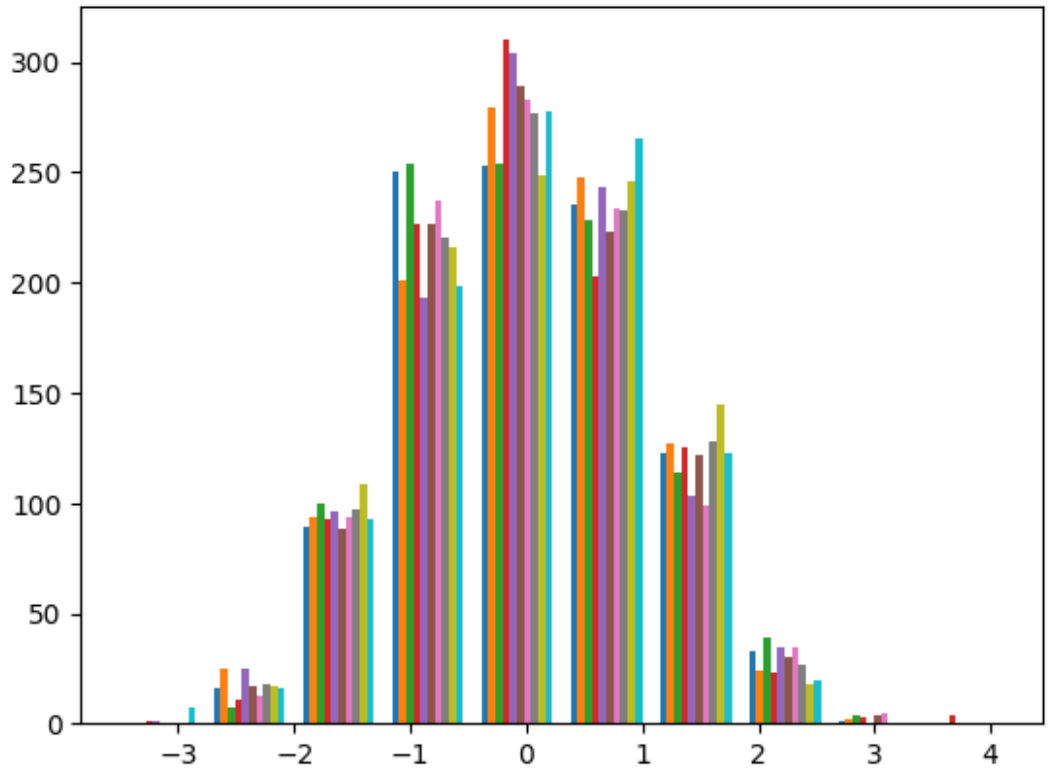

```
{"summary":"{\n  \"name\": \"df_feat\",\n  \"rows\": 1000,\n
\"fields\": [\n    {\n      \"column\": \"WTT\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.000500375312774,\n        \"min\": -2.678050294892763,\n
\"max\": 2.667092453480776,\n        \"num_unique_values\": 1000,\n
\"samples\": [\n          -0.8407204008480645,\n
1.3816554504162177,\n          1.2685017266467709\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"PTI\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1.0005003753127737,\n
\"min\": -2.6187465594530646,\n        \"max\": 2.799903822014538,\n
\"num_unique_values\": 1000,\n        \"samples\": [\n
1.5834383679497588,\n          0.41264176640541583,\n
0.13366769728284045\n        ],\n        \"semantic_type\": \"\",\n
```

```
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"EQW\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1.0005003753127737,\n        \"min\": -2.2758578511888814,\n
\"max\": 3.049325158601135,\n        \"num_unique_values\": 1000,\n
\"samples\": [\n          1.0180932040839803,\n          -
0.4401552303536436,\n          0.7178222625994329\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"SBI\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1.0005003753127737,\n
\"min\": -2.775551017934635,\n        \"max\": 4.151021461563324,\n
\"num_unique_values\": 1000,\n        \"samples\": [\n          -
1.2140483883610458,\n          0.576212065632318,\n
0.9576951405925229\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"LQE\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1.0005003753127737,\n        \"min\": -2.947205948503044,\n
\"max\": 2.538987100864661,\n        \"num_unique_values\": 1000,\n
\"samples\": [\n          -0.5573184627252994,\n          -
1.4896520988184299,\n          -0.3616080116045524\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"QWG\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1.0005003753127735,\n
\"min\": -2.660802498981584,\n        \"max\": 2.8257390273790977,\n
\"num_unique_values\": 1000,\n        \"samples\": [\n          -
0.1081173689453796,\n          -1.060712948686669,\n
0.6454747043228094\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"FDJ\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1.0005003753127737,\n        \"min\": -2.6204660364684953,\n
\"max\": 2.940974438292232,\n        \"num_unique_values\": 1000,\n
\"samples\": [\n          0.4185386031864608,\n          -
0.8497069652008912,\n          -2.5498213156881664\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"PJF\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1.0005003753127737,\n
\"min\": -2.6744652897720127,\n        \"max\": 2.4701088805242493,\n
\"num_unique_values\": 1000,\n        \"samples\": [\n
0.4953379133015279 3,\n          -1.8392486714127034,\n          -
0.9045163698230286\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"HQE\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1.000500375312774,\n        \"min\": -2.7013608425848443,\n
\"max\": 2.4777335068709863,\n        \"num_unique_values\": 1000,\n
\"samples\": [\n          -0.4078671888285603,\n
0.9323937672052752,\n          -0.422001648049673\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"NXJ\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1.000500375312774,\n
\"min\": -3.542140064466624,\n        \"max\": 2.6024764661358253,\n
```

```
\"num_unique_values\": 1000,\n        \"samples\": [\n
0.40697451922755945,\n          -0.9287136944022932,\n
1.0896815016660177\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df_feat"}
```

```
plt.hist(df_feat)
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```




```
from sklearn.model_selection import train_test_split
X = df_feat
y = df['TARGET CLASS']
X_train, X_test, y_train, y_test = train_test_split(scaled_features,
df['TARGET CLASS'], test_size = 0.30, random_state = 101)

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 1)
knn.fit(X_train, y_train)

KNeighborsClassifier(n_neighbors=1)

pred = knn.predict(X_test)
```

```
from sklearn.metrics import classification_report, confusion_matrix
conf_mat = confusion_matrix(y_test,pred)
print(conf_mat)
```

```
[[151   8]
 [ 15 126]]
```

```
print(classification_report(y_test,pred))
```

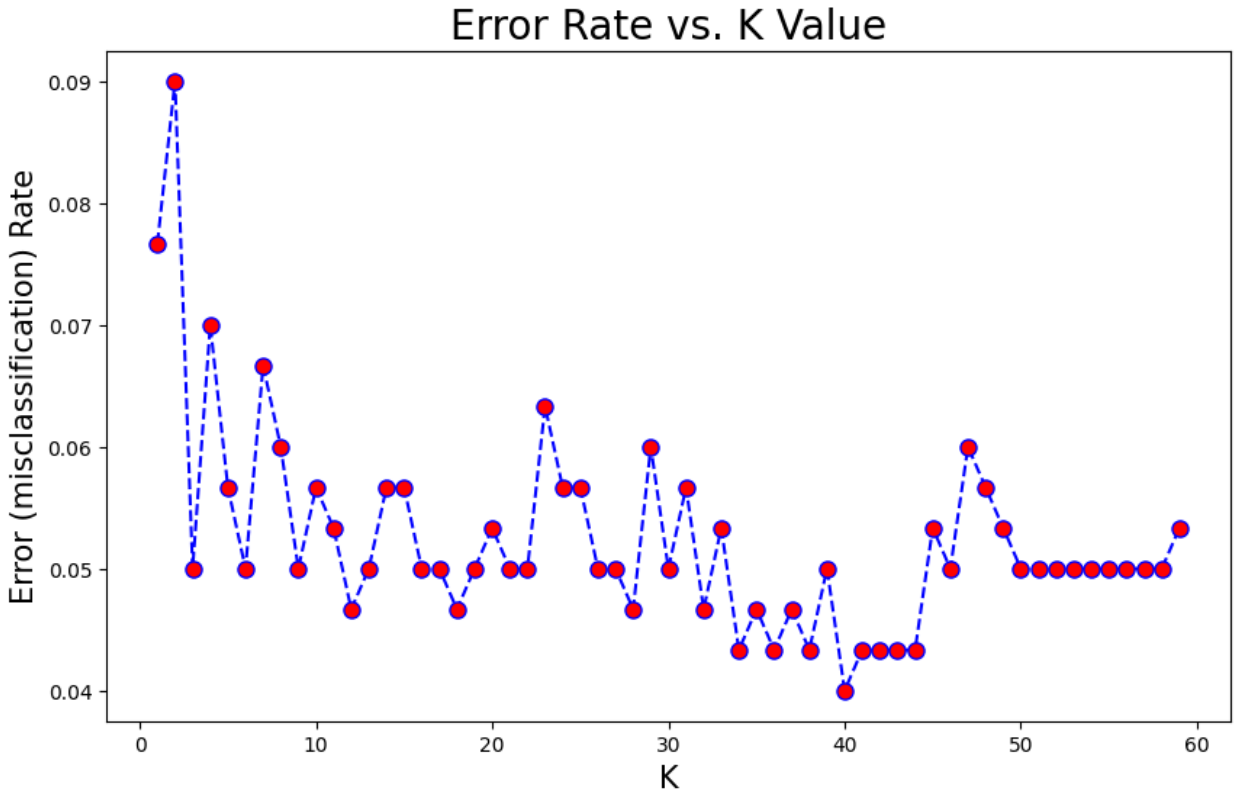
```
              precision    recall  f1-score   support

           0       0.91      0.95      0.93       159
           1       0.94      0.89      0.92       141

    accuracy                           0.92       300
   macro avg       0.92      0.92      0.92       300
weighted avg       0.92      0.92      0.92       300
```

```
print("Missclassification error rate: ", round(np.mean(pred!=y_test),
3))
```

```
Missclassification error rate:  0.077
```

```
error_rate = []
for i in range(1,60):
  knn = KNeighborsClassifier(n_neighbors = i)
  knn.fit(X_train, y_train)
  pred_i = knn.predict(X_test)
  error_rate.append(np.mean(pred_i != y_test))

plt.figure(figsize = (10,6))
plt.plot(range(1,60),error_rate,color='blue', linestyle='dashed',
marker = 'o', markerfacecolor='red', markersize=8)
plt.title('Error Rate vs. K Value', fontsize = 20)
plt.xlabel('K', fontsize=15)
plt.ylabel('Error (misclassification) Rate', fontsize=15)
```

```
Text(0, 0.5, 'Error (misclassification) Rate')
```


Error Rate vs. K Value
Error (misclassification) Rate
0.09
0.08
0.07
0.06
0.05
0.04
0
10
20
30
40
50
60
K