In [1]:
```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import zscore
from sklearn.preprocessing import LabelEncoder
from sklearn.cluster import KMeans
import plotly.express as px
```

In [2]:
```
df=pd.read_csv('temperature_device_failure2.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	timestamp	value
0	2013-07-04 00:00:00	69.880835
1	2013-07-04 01:00:00	71.220227
2	2013-07-04 02:00:00	70.877805
3	2013-07-04 03:00:00	68.959400
4	2013-07-04 04:00:00	69.283551

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
timestamp    0
value        0
dtype: int64
```

In [5]:
```
df.dtypes
```

Out[5]:
```
timestamp     object
value        float64
dtype: object
```

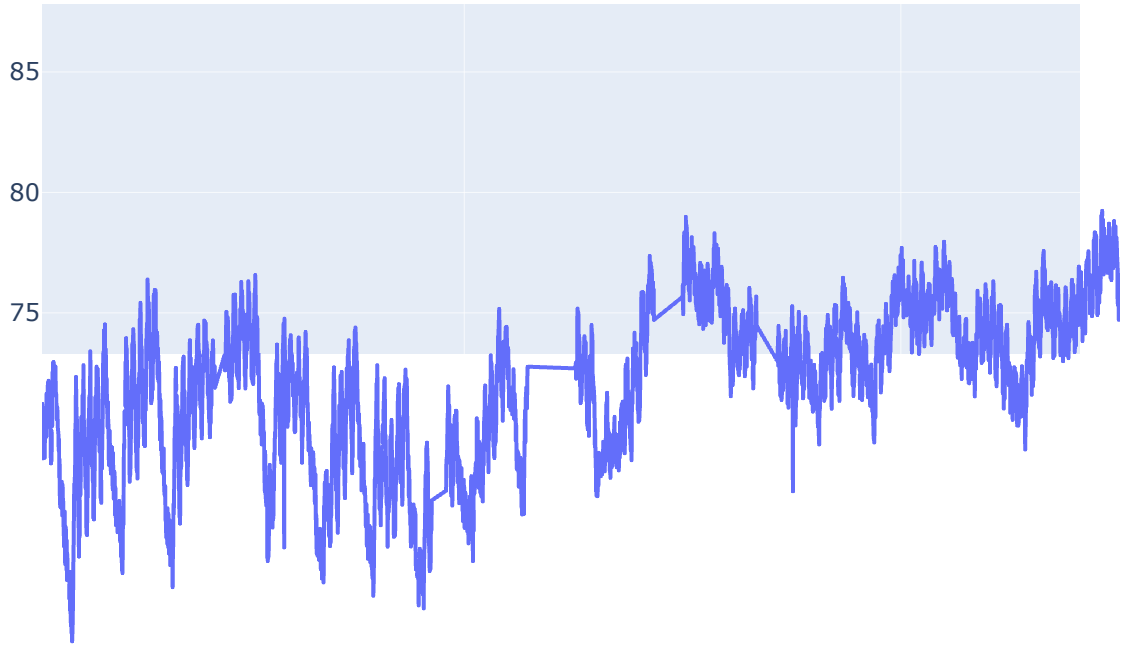
In [6]:
```
df.describe()
```

Out[6]:

	value
count	7267.000000
mean	71.242433
std	4.247509
min	57.458406
25%	68.369411
50%	71.858493
75%	74.430958
max	86.223213

DATA VISUALIZATION

In [7]:
```
px.line(x=df['timestamp'],y=df['value'])
```



Extracting Feature

In [9]:
```
df1=df.copy()
```

In [10]:
```
df1['timestamp'] = pd.to_datetime(df1['timestamp'])
```

In [11]:
```
df1.dtypes
```

Out[11]:
```
timestamp    datetime64[ns]
value               float64
dtype: object
```

In [12]:
```
df1['dayofweek'] = df1['timestamp'].dt.day_name()
df1
```

Out[12]:

	timestamp	value	dayofweek
0	2013-07-04 00:00:00	69.880835	Thursday
1	2013-07-04 01:00:00	71.220227	Thursday
2	2013-07-04 02:00:00	70.877805	Thursday
3	2013-07-04 03:00:00	68.959400	Thursday
4	2013-07-04 04:00:00	69.283551	Thursday
...	...	...	...
7262	2014-05-28 11:00:00	72.370206	Wednesday
7263	2014-05-28 12:00:00	72.172956	Wednesday
7264	2014-05-28 13:00:00	72.046565	Wednesday
7265	2014-05-28 14:00:00	71.825226	Wednesday
7266	2014-05-28 15:00:00	72.584089	Wednesday

7267 rows × 3 columns

In [13]:
```
def days_type(day):
    day_type=''

    if day=='Monday' or day=='Tuesday' or day=='Wednesday' or day=='Thursday' or day==
        day_type='Weekday'
    else:
        day_type='Weekend'
    return day_type

df1['day_type']=df1['dayofweek'].apply(days_type)
df1.head()
```

Out[13]:

	timestamp	value	dayofweek	day_type
0	2013-07-04 00:00:00	69.880835	Thursday	Weekday
1	2013-07-04 01:00:00	71.220227	Thursday	Weekday
2	2013-07-04 02:00:00	70.877805	Thursday	Weekday
3	2013-07-04 03:00:00	68.959400	Thursday	Weekday
4	2013-07-04 04:00:00	69.283551	Thursday	Weekday

In [14]:
```
df1[df1['day_type']=='Weekend'].head()
```

Out[14]:

	timestamp	value	dayofweek	day_type
48	2013-07-06 00:00:00	71.630964	Saturday	Weekend
49	2013-07-06 01:00:00	70.596735	Saturday	Weekend
50	2013-07-06 02:00:00	70.852482	Saturday	Weekend
51	2013-07-06 03:00:00	71.084768	Saturday	Weekend
52	2013-07-06 04:00:00	70.847233	Saturday	Weekend

In [15]:

```
def day_night(x):
    d,t=x.split(' ')
    h,m=map(int,t.split(':'))
    if h>=7 and h<=19:
        return 'Day'
    else:
        return 'Night'
```

In [16]:

```
def categorize_time(timestamp):
    hour = timestamp.hour
    if 7 <= hour < 19:
        if timestamp.weekday() < 5:
            return 'Weekday day'
        else:
            return 'Weekend day'
    else:
        if timestamp.weekday() < 5:
            return 'Weekday night'
        else:
            return 'Weekend night'
df1['DTC'] = df1['timestamp'].apply(categorize_time)
df1
```

Out[16]:

	timestamp	value	dayofweek	day_type	DTC
0	2013-07-04 00:00:00	69.880835	Thursday	Weekday	Weekday night
1	2013-07-04 01:00:00	71.220227	Thursday	Weekday	Weekday night
2	2013-07-04 02:00:00	70.877805	Thursday	Weekday	Weekday night
3	2013-07-04 03:00:00	68.959400	Thursday	Weekday	Weekday night
4	2013-07-04 04:00:00	69.283551	Thursday	Weekday	Weekday night
...	...	...	...	...	...
7262	2014-05-28 11:00:00	72.370206	Wednesday	Weekday	Weekday day
7263	2014-05-28 12:00:00	72.172956	Wednesday	Weekday	Weekday day
7264	2014-05-28 13:00:00	72.046565	Wednesday	Weekday	Weekday day
7265	2014-05-28 14:00:00	71.825226	Wednesday	Weekday	Weekday day
7266	2014-05-28 15:00:00	72.584089	Wednesday	Weekday	Weekday day

7267 rows × 5 columns

In [17]:
```
df2=df1.drop(columns=['timestamp','dayofweek','day_type'], axis=1).copy()
df2.head()
```

Out[17]:

	value	DTC
0	69.880835	Weekday night
1	71.220227	Weekday night
2	70.877805	Weekday night
3	68.959400	Weekday night
4	69.283551	Weekday night

In [18]:
```
LE_DTC = LabelEncoder()
df2['DTC'] = LE_DTC.fit_transform(df2['DTC'])
LE_DTC.classes_
```

Out[18]:
```
array(['Weekday day', 'Weekday night', 'Weekend day', 'Weekend night'],
      dtype=object)
```

In [19]:
```
df2
```

Out[19]:

	value	DTC
0	69.880835	1
1	71.220227	1
2	70.877805	1
3	68.959400	1
4	69.283551	1
...	...	...
7262	72.370206	0
7263	72.172956	0
7264	72.046565	0
7265	71.825226	0
7266	72.584089	0

7267 rows × 2 columns

In [20]:
```
df1['DTC1']=df2['DTC'].copy()
df1
```

Out[20]:

	timestamp	value	dayofweek	day_type	DTC	DTC1
0	2013-07-04 00:00:00	69.880835	Thursday	Weekday	Weekday night	1
1	2013-07-04 01:00:00	71.220227	Thursday	Weekday	Weekday night	1
2	2013-07-04 02:00:00	70.877805	Thursday	Weekday	Weekday night	1
3	2013-07-04 03:00:00	68.959400	Thursday	Weekday	Weekday night	1
4	2013-07-04 04:00:00	69.283551	Thursday	Weekday	Weekday night	1
...	...	...	...	...	...	...
7262	2014-05-28 11:00:00	72.370206	Wednesday	Weekday	Weekday day	0
7263	2014-05-28 12:00:00	72.172956	Wednesday	Weekday	Weekday day	0
7264	2014-05-28 13:00:00	72.046565	Wednesday	Weekday	Weekday day	0
7265	2014-05-28 14:00:00	71.825226	Wednesday	Weekday	Weekday day	0
7266	2014-05-28 15:00:00	72.584089	Wednesday	Weekday	Weekday day	0

7267 rows × 6 columns

In [21]:
```
df1[df1['DTC1']==0].head()
```

Out[21]:

	timestamp	value	dayofweek	day_type	DTC	DTC1
7	2013-07-04 07:00:00	69.369608	Thursday	Weekday	Weekday day	0
8	2013-07-04 08:00:00	69.166714	Thursday	Weekday	Weekday day	0
9	2013-07-04 09:00:00	68.986083	Thursday	Weekday	Weekday day	0
10	2013-07-04 10:00:00	69.965062	Thursday	Weekday	Weekday day	0
11	2013-07-04 11:00:00	70.556195	Thursday	Weekday	Weekday day	0

In [22]:
```
df1[df1['DTC1']==2].head()
```

Out[22]:

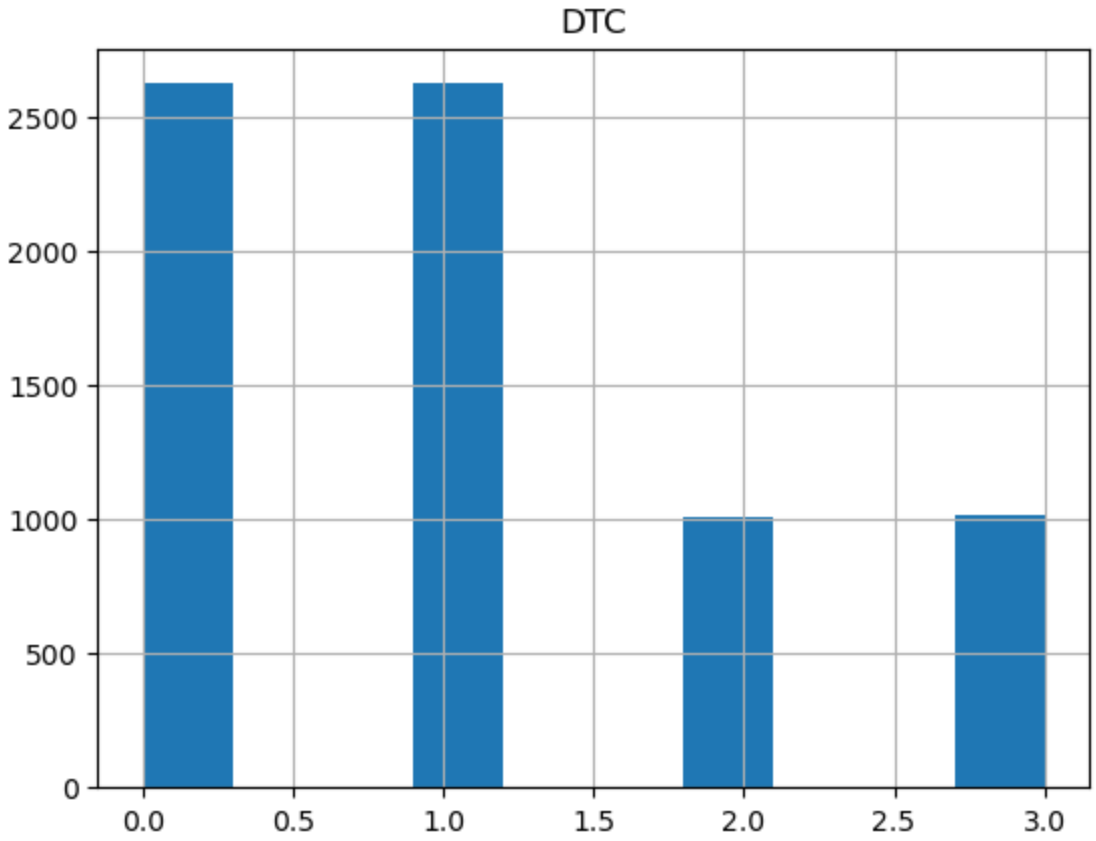
	timestamp	value	dayofweek	day_type	DTC	DTC1
55	2013-07-06 07:00:00	69.929231	Saturday	Weekend	Weekend day	2
56	2013-07-06 08:00:00	69.285750	Saturday	Weekend	Weekend day	2
57	2013-07-06 09:00:00	69.726387	Saturday	Weekend	Weekend day	2
58	2013-07-06 10:00:00	68.190103	Saturday	Weekend	Weekend day	2
59	2013-07-06 11:00:00	68.916795	Saturday	Weekend	Weekend day	2

Ultimately, we would like to figure out when (weekday, weekend, day or night) the device fails!

In [23]:
```
df3=df2.copy()
```

In [24]:
```
df3.hist(column='DTC')
```

Out[24]:
```
array([[<Axes: title={'center': 'DTC'}>]], dtype=object)
```



In [25]:
```
df3.head()
```

Out[25]:

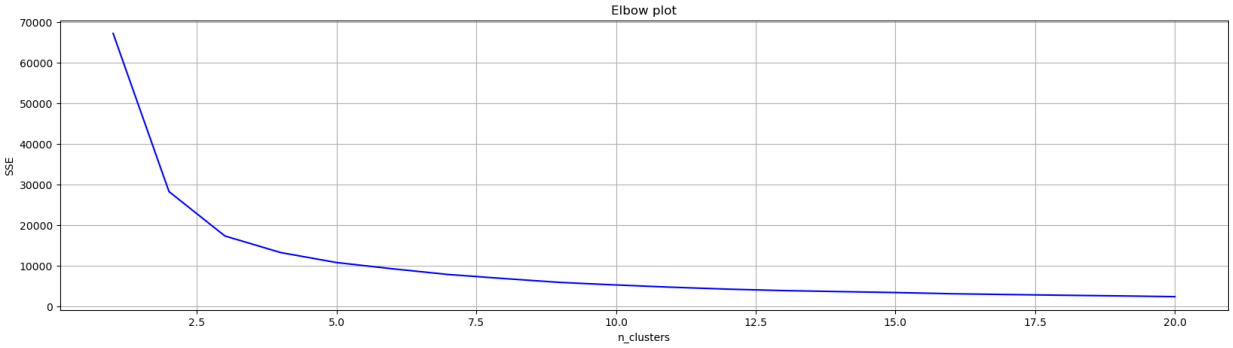
	value	DTC
0	69.880835	1
1	71.220227	1
2	70.877805	1
3	68.959400	1
4	69.283551	1

In [26]:
```
split=int(df3.values.shape[0]*0.75)
train=df3.values[:split,:]
test=df3.values[split:,:]
sse=[]
for i in range(1,21):
    km=KMeans(n_clusters=i).fit(train)
    sse.append(km.inertia_)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
```

In [27]:
```
plt.figure(figsize=(20,5))
plt.plot([x for x in range(1,21)],sse,color='b')
plt.xlabel('n_clusters')
plt.ylabel('SSE')
plt.title('Elbow plot')
plt.grid()
plt.show()
```



In [28]:
```
df3['labels']=KMeans(n_clusters=3).fit_predict(df2.values)
df3
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of
`n_init` explicitly to suppress the warning
```

Out[28]:

	value	DTC	labels
0	69.880835	1	2
1	71.220227	1	2
2	70.877805	1	2
3	68.959400	1	2
4	69.283551	1	2
...	...	...	...
7262	72.370206	0	2
7263	72.172956	0	2
7264	72.046565	0	2
7265	71.825226	0	2
7266	72.584089	0	2

7267 rows × 3 columns

In [29]:
```
#Weekday Day
print(df3.loc[df3['labels']==0].describe())
```

```
              value          DTC  labels
count  2697.000000  2697.000000  2697.0
mean     75.396785     0.971079     0.0
std       1.747844     0.966759     0.0
min      73.130321     0.000000     0.0
25%      74.090658     0.000000     0.0
50%      75.111465     1.000000     0.0
75%      76.213956     1.000000     0.0
max      86.223213     3.000000     0.0
```

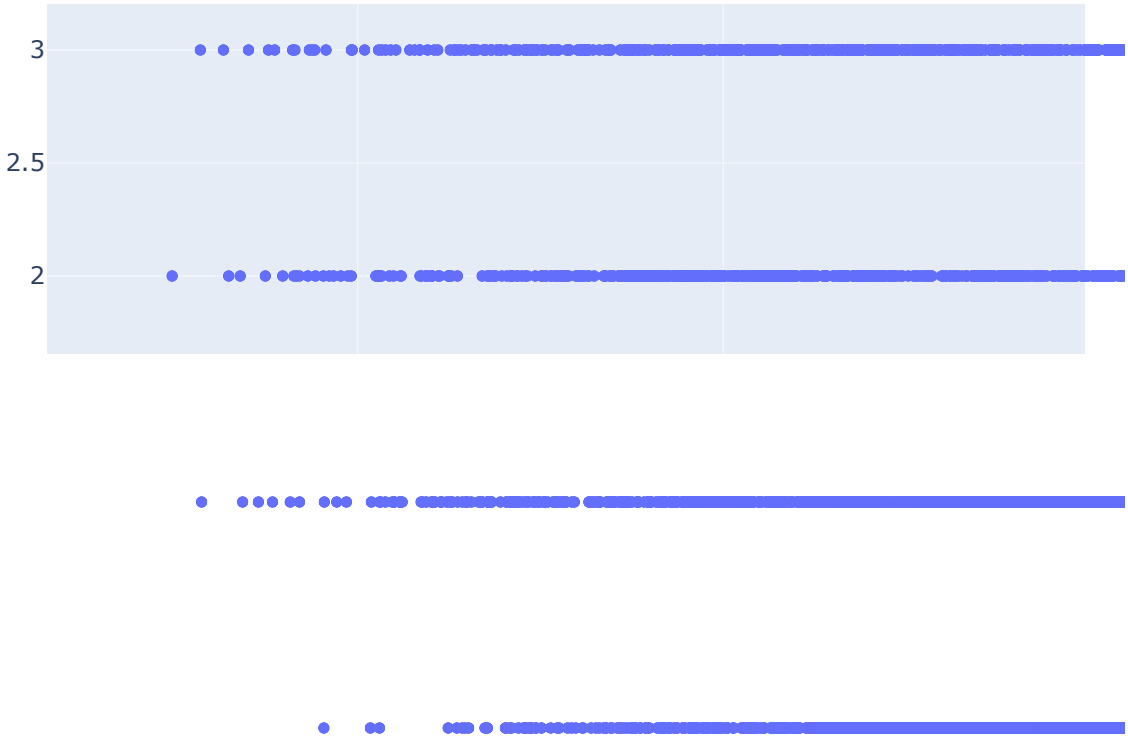
In [30]:
```
#Weekend Day
print(df3.loc[df3['labels']==2].describe())
```

```
              value          DTC  labels
count  2913.000000  2913.000000  2913.0
mean     70.882134     0.904566     2.0
std       1.469123     0.997502     0.0
min      67.893469     0.000000     2.0
25%      69.725078     0.000000     2.0
50%      71.019301     1.000000     2.0
75%      72.177081     1.000000     2.0
max      73.158129     3.000000     2.0
```

In [31]:
```
#weekday night
print(df3.loc[df3['labels']==1].describe())
```

```
              value          DTC  labels
count  1657.000000  1657.000000  1657.0
mean     65.114046     1.467109     1.0
std       2.115376     1.067317     0.0
min      57.458406     0.000000     1.0
25%      63.918387     1.000000     1.0
50%      65.504216     1.000000     1.0
75%      66.797619     2.000000     1.0
max      68.178555     3.000000     1.0
```

In [32]:
```
fig=px.scatter(df3, x='value', y='DTC')
fig.show()
print(df3.head())
```



	value	DTC	labels
0	69.880835	1	2
1	71.220227	1	2
2	70.877805	1	2
3	68.959400	1	2
4	69.283551	1	2

Principal Components

In [33]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pcadf=pca.fit_transform(train)
print(pca.explained_variance_ratio_)
```

```
[0.91554867 0.08445133]
```

In [34]:
```
PCA= pd.DataFrame(data = pcadf, columns = ['pca1', 'pca2'])
PCA
```

Out[34]:

	pca1	pca2
0	-2.872507	-0.133705
1	-1.533699	-0.094146
2	-1.875972	-0.104260
3	-3.793540	-0.160920
4	-3.469530	-0.151346
...	...	...
5445	-1.399272	-0.090174
5446	-2.362463	-0.118634
5447	-3.315024	-0.146781
5448	-2.495013	-0.122551
5449	-3.081348	-0.139876

5450 rows × 2 columns

In [35]:
```
PCA['DTC']=df1['DTC'].copy()
PCA
```

Out[35]:

	pca1	pca2	DTC
0	-2.872507	-0.133705	Weekday night
1	-1.533699	-0.094146	Weekday night
2	-1.875972	-0.104260	Weekday night
3	-3.793540	-0.160920	Weekday night
4	-3.469530	-0.151346	Weekday night
...	...	...	...
5445	-1.399272	-0.090174	Weekday night
5446	-2.362463	-0.118634	Weekday night
5447	-3.315024	-0.146781	Weekday night
5448	-2.495013	-0.122551	Weekday night
5449	-3.081348	-0.139876	Weekday night

5450 rows × 3 columns

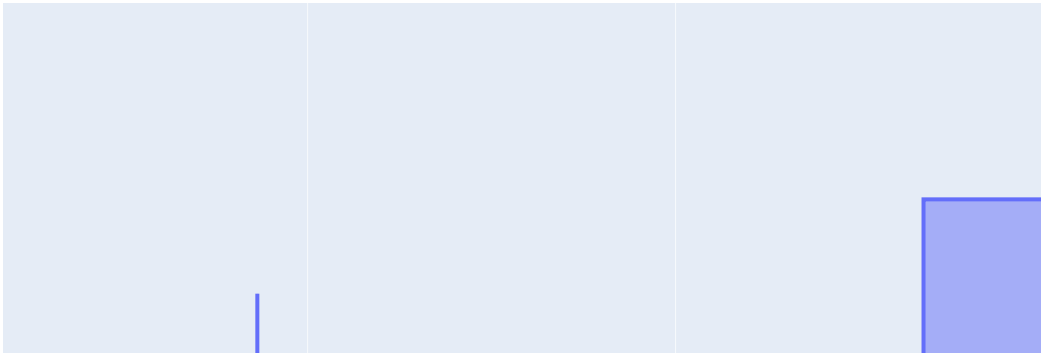
OUTLIERS

In [36]:
```
Q1=df3.value.quantile(.25)
Q3=df3.value.quantile(.75)
IQR=Q3-Q1
lower_limit=Q1-(1.5*IQR)
```

```
upper_limit=Q3+(1.5*IQR)
print("Min Value",df3.value.min())
print("Max Value",df3.value.max())
print('Q1',Q1)
print('Q3',Q3)
print('IQR',IQR)
print('lower Limit',lower_limit)
print('upper Limit',upper_limit)
```

```
Min Value 57.45840559
Max Value 86.22321261
Q1 68.36941051
Q3 74.43095786
IQR 6.061547350000012
lower Limit 59.277089484999976
upper Limit 83.52327888500002
```

In [37]:
```
px.box(df3,x='value')
```



In [38]:
```
outliers=df2.copy()
```

In [39]:
```
upperlimit, lowerlimit = outliers.value.quantile([0.99,0.01])
upperlimit, lowerlimit
```

Out[39]:
```
(79.219261534, 60.847063600599995)
```

In [40]:
```
outliers1 = pd.DataFrame(outliers[(outliers['value']<upperlimit) & (outliers['value']>
print(len(outliers))
print(len(outliers1))
```

```
7267
7121
```

In [41]:
```
print("Outliers Removed :\n",len(outliers)-len(outliers1))
```

```
Outliers Removed :
 146
```

EllipticEnvelope

In [42]:
```
from sklearn.covariance import EllipticEnvelope
```

In [43]:
```
EE = df2.copy()
```

In [44]:
```
EE1 = EllipticEnvelope(contamination = 0.01).fit(EE[['value']])
```

In [45]:
```
EE['score']=EE1.decision_function(EE[['value']])
EE
```

Out[45]:

	value	DTC	score
0	69.880835	1	9.571927
1	71.220227	1	9.814768
2	70.877805	1	9.778707
3	68.959400	1	9.246070
4	69.283551	1	9.375461
...	...	...	...
7262	72.370206	0	9.805053
7263	72.172956	0	9.821045
7264	72.046565	0	9.828173
7265	71.825226	0	9.834790
7266	72.584089	0	9.781011

7267 rows × 3 columns

In [46]:
```
EE['outlier']=EE1.predict(EE[['value']])
EE
```

Out[46]:

	value	DTC	score	outlier
0	69.880835	1	9.571927	1
1	71.220227	1	9.814768	1
2	70.877805	1	9.778707	1
3	68.959400	1	9.246070	1
4	69.283551	1	9.375461	1
...	...	...	...	...
7262	72.370206	0	9.805053	1
7263	72.172956	0	9.821045	1
7264	72.046565	0	9.828173	1
7265	71.825226	0	9.834790	1
7266	72.584089	0	9.781011	1

7267 rows × 4 columns

In [49]:
```
EE.loc[EE['outlier']==1]
```

Out[49]:

	value	DTC	score	outlier
0	69.880835	1	9.571927	1
1	71.220227	1	9.814768	1
2	70.877805	1	9.778707	1
3	68.959400	1	9.246070	1
4	69.283551	1	9.375461	1
...	...	...	...	...
7262	72.370206	0	9.805053	1
7263	72.172956	0	9.821045	1
7264	72.046565	0	9.828173	1
7265	71.825226	0	9.834790	1
7266	72.584089	0	9.781011	1

7194 rows × 4 columns

In [50]:
```
EE.loc[EE['outlier']==-1].copy()
```

Out[50]:

	value	DTC	score	outlier
3699	83.247886	3	-0.259722	-1
3702	83.780995	3	-1.216688	-1
3717	83.511630	2	-0.727744	-1
3718	84.390932	2	-2.364710	-1
3719	85.227685	2	-4.031919	-1
...	...	...	...	...
7038	59.074691	1	-2.391697	-1
7040	59.711858	1	-1.192387	-1
7041	60.375894	1	-0.008366	-1
7042	60.171092	0	-0.366371	-1
7043	60.296682	0	-0.146074	-1

73 rows × 4 columns

In [52]:
```
A = len(EE[EE['value'] > 79.219261534])
accuracy = 100*list(EE['outlier']).count(-1)/(c)
print("Accuracy of the model:", accuracy)
A
```

Out[52]:
```
Accuracy of the model: 100.0
73
```

In [54]:
```
A1 = len(EE[EE['value'] > 79.219261534])
A2 = len(EE[EE['value'] < 60.847063600599995])
A=A1+A2
accuracy = 100*list(EE['outlier']).count(-1)/(c)
print("Accuracy of the model:", accuracy)
A
```

Out[54]:
```
Accuracy of the model: 100.0
146
```

Isolation Forest

In [55]:
```
from sklearn.ensemble import IsolationForest
ISO= df2.copy()
```

In [56]:
```
IsoFor =IsolationForest(n_estimators=100, max_samples='auto', contamination= 0.01,max_
```

In [57]:
```
ISO['anomaly_score']=IsoFor.decision_function(ISO[['value']])
ISO
```

Out[57]:

	value	DTC	anomaly_score
0	69.880835	1	0.254169
1	71.220227	1	0.264449
2	70.877805	1	0.262367
3	68.959400	1	0.241853
4	69.283551	1	0.243933
...	...	...	...
7262	72.370206	0	0.260305
7263	72.172956	0	0.265659
7264	72.046565	0	0.264874
7265	71.825226	0	0.267483
7266	72.584089	0	0.269801

7267 rows × 3 columns

In [58]:
```
ISO['outlier']=IsoFor.predict(ISO[['value']])
ISO
```

Out[58]:

	value	DTC	anomaly_score	outlier
0	69.880835	1	0.254169	1
1	71.220227	1	0.264449	1
2	70.877805	1	0.262367	1
3	68.959400	1	0.241853	1
4	69.283551	1	0.243933	1
...	...	...	...	...
7262	72.370206	0	0.260305	1
7263	72.172956	0	0.265659	1
7264	72.046565	0	0.264874	1
7265	71.825226	0	0.267483	1
7266	72.584089	0	0.269801	1

7267 rows × 4 columns

In [59]:
```
ISO.loc[ISO['outlier']==1]
```

Out[59]:

	value	DTC	anomaly_score	outlier
0	69.880835	1	0.254169	1
1	71.220227	1	0.264449	1
2	70.877805	1	0.262367	1
3	68.959400	1	0.241853	1
4	69.283551	1	0.243933	1
...	...	...	...	...
7262	72.370206	0	0.260305	1
7263	72.172956	0	0.265659	1
7264	72.046565	0	0.264874	1
7265	71.825226	0	0.267483	1
7266	72.584089	0	0.269801	1

7194 rows × 4 columns

In [60]:
```
ISO.loc[ISO['outlier']==-1]
```

Out[60]:

	value	DTC	anomaly_score	outlier
3697	82.289240	3	-0.033552	-1
3698	82.989869	3	-0.048516	-1
3699	83.247886	3	-0.053220	-1
3700	82.519659	3	-0.038165	-1
3701	82.736802	3	-0.043841	-1
...	...	...	...	...
7034	58.160342	3	-0.025834	-1
7035	58.423639	1	-0.025834	-1
7036	57.861906	1	-0.027353	-1
7037	58.639295	1	-0.023310	-1
7038	59.074691	1	-0.019288	-1

73 rows × 4 columns

In [61]:
```
c = len(ISO[ISO['value'] > 79.219261534])
accuracy_iso = 100*list(ISO['outlier']).count(-1)/(c)
print("Accuracy of the model:", accuracy_iso)
c
```

```
Accuracy of the model: 100.0
73
```

Out[61]:

In [62]:
```
A1 = len(ISO[ISO['value'] > 79.219261534])
A2 = len(ISO[ISO['value'] < 60.847063600599995])
A=A1+A2
accuracy_iso = 100*list(ISO['outlier']).count(-1)/(c)
print("Accuracy of the model:", accuracy_iso)
c
```

```
Accuracy of the model: 100.0
73
```

Out[62]:

In [63]:
```
df1
```

Out[63]:

	timestamp	value	dayofweek	day_type	DTC	DTC1
0	2013-07-04 00:00:00	69.880835	Thursday	Weekday	Weekday night	1
1	2013-07-04 01:00:00	71.220227	Thursday	Weekday	Weekday night	1
2	2013-07-04 02:00:00	70.877805	Thursday	Weekday	Weekday night	1
3	2013-07-04 03:00:00	68.959400	Thursday	Weekday	Weekday night	1
4	2013-07-04 04:00:00	69.283551	Thursday	Weekday	Weekday night	1
...	...	...	...	...	...	...
7262	2014-05-28 11:00:00	72.370206	Wednesday	Weekday	Weekday day	0
7263	2014-05-28 12:00:00	72.172956	Wednesday	Weekday	Weekday day	0
7264	2014-05-28 13:00:00	72.046565	Wednesday	Weekday	Weekday day	0
7265	2014-05-28 14:00:00	71.825226	Wednesday	Weekday	Weekday day	0
7266	2014-05-28 15:00:00	72.584089	Wednesday	Weekday	Weekday day	0

7267 rows × 6 columns

In [65]:
```
TD=df1.drop(columns=['timestamp','dayofweek','DTC'], axis=1).copy()
TD.head(15)
```

Out[65]:

	value	day_type	DTC1
0	69.880835	Weekday	1
1	71.220227	Weekday	1
2	70.877805	Weekday	1
3	68.959400	Weekday	1
4	69.283551	Weekday	1
5	70.060966	Weekday	1
6	69.279765	Weekday	1
7	69.369608	Weekday	0
8	69.166714	Weekday	0
9	68.986083	Weekday	0
10	69.965062	Weekday	0
11	70.556195	Weekday	0
12	70.307505	Weekday	0
13	70.246252	Weekday	0
14	69.854908	Weekday	0

In [67]:
```
LE_dtype = LabelEncoder()
TD['day_type'] = LE_DTC.fit_transform(TD['day_type'])
LE_DTC.classes_
TD.head()
```

Out[67]:

	value	day_type	DTC1
0	69.880835	0	1
1	71.220227	0	1
2	70.877805	0	1
3	68.959400	0	1
4	69.283551	0	1

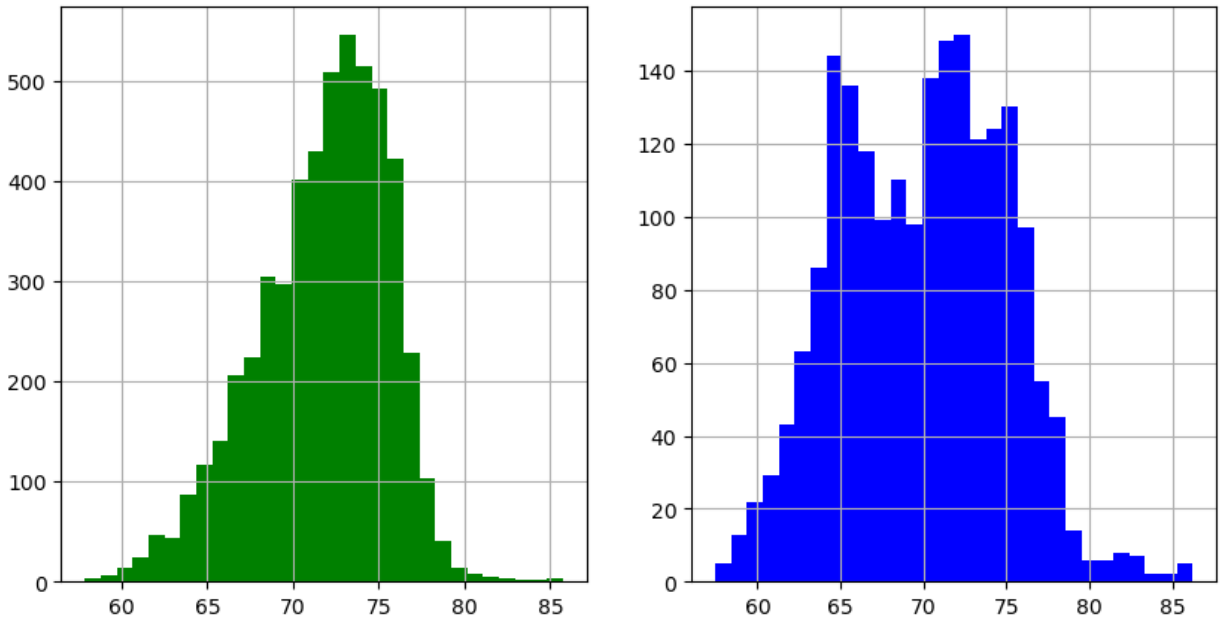
In [68]:
```
Ul=79.219261534
Ll=60.847063600599995
```

In [69]:
```
TD2=TD.copy()
TD2[(TD2.value < Ll) | (TD2.value > Ul)].count()
```

Out[69]:
```
value       146
day_type    146
DTC1        146
dtype: int64
```

In [70]:
```
weekday = TD.loc[TD['day_type'] == 0, 'value']
weekend = TD.loc[TD['day_type'] == 1, 'value']
```

```
fig, axs = plt.subplots(1,2, figsize=(10, 5))
weekday.hist(ax=axs[0], bins=30, color='green')
weekend.hist(ax=axs[1], bins=30, color='blue');
```

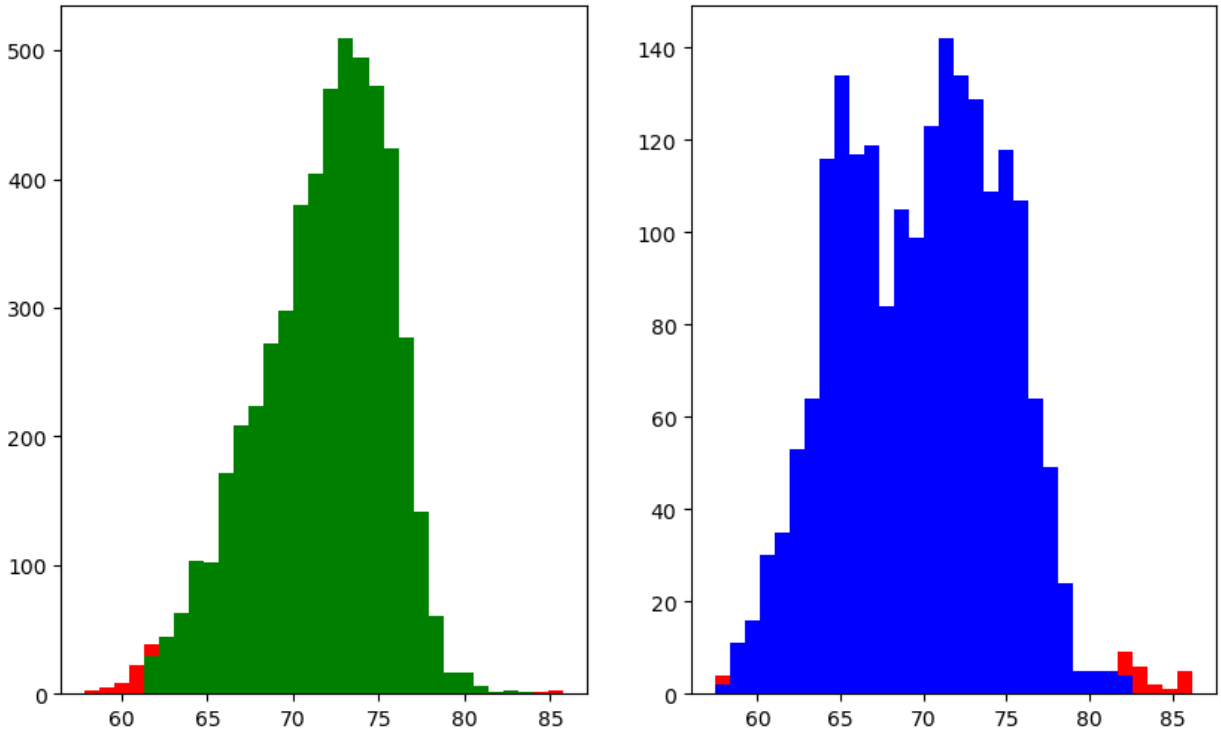


EDA(EllipticEnvelope)

In [71]:
```
envelope =  EllipticEnvelope(contamination = 0.01)
X_train = weekday.values.reshape(-1,1)
envelope.fit(X_train)
week_day = pd.DataFrame(weekday)
week_day['score'] = envelope.decision_function(X_train)
week_day['anomaly'] = envelope.predict(X_train)

envelope =  EllipticEnvelope(contamination = 0.01)
X_train = weekend.values.reshape(-1,1)
envelope.fit(X_train)
week_end = pd.DataFrame(weekend)
week_end['score'] = envelope.decision_function(X_train)
week_end['anomaly'] = envelope.predict(X_train)
```

In [73]:
```
# plot the price repartition by categories with anomalies
D0 = week_day.loc[week_day['anomaly'] == 1, 'value']
E0 = week_day.loc[week_day['anomaly'] == -1, 'value']

D1 = week_end.loc[week_end['anomaly'] == 1, 'value']
E1 = week_end.loc[week_end['anomaly'] == -1, 'value']

fig, axs = plt.subplots(1,2, figsize=(10, 6))
axs[0].hist([D0,E0], bins=32, stacked=True, color=['green', 'red'])
axs[1].hist([D1,E1], bins=32, stacked=True, color=['blue', 'red'])
```

Out[73]:
```
(array([[  2.,  11.,  16.,  30.,  35.,  53.,  64., 116., 134., 117., 119.,
         84., 105.,  99., 123., 142., 134., 129., 109., 118., 107.,  64.,
         49.,  24.,   5.,   5.,   5.,   4.,   0.,   0.,   0.,   0.],
        [  4.,  11.,  16.,  30.,  35.,  53.,  64., 116., 134., 117., 119.,
         84., 105.,  99., 123., 142., 134., 129., 109., 118., 107.,  64.,
         49.,  24.,   5.,   5.,   5.,   9.,   6.,   2.,   1.,   5.]]),
 array([57.45840559, 58.35730581, 59.25620603, 60.15510625, 61.05400647,
        61.95290669, 62.85180691, 63.75070713, 64.64960734, 65.54850756,
        66.44740778, 67.346308  , 68.24520822, 69.14410844, 70.04300866,
        70.94190888, 71.8408091 , 72.73970932, 73.63860954, 74.53750976,
        75.43640998, 76.3353102 , 77.23421042, 78.13311064, 79.03201086,
        79.93091107, 80.82981129, 81.72871151, 82.62761173, 83.52651195,
        84.42541217, 85.32431239, 86.22321261]),
 <a list of 2 BarContainer objects>)
```



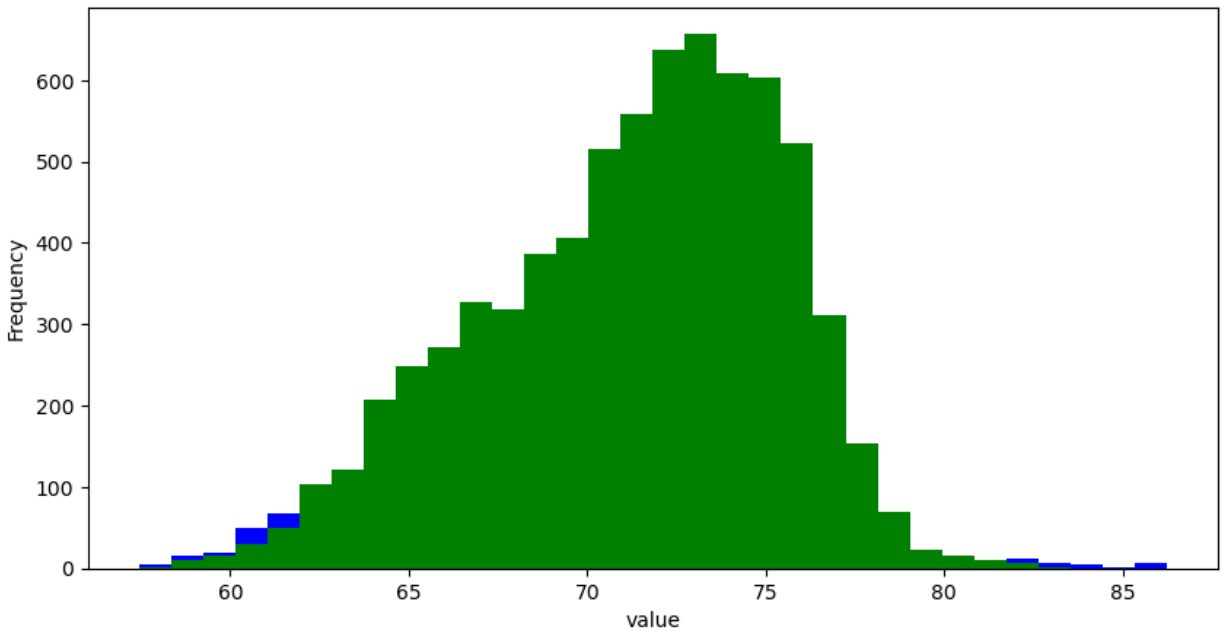
In [74]:
```
Day_End = pd.concat([week_day, week_end])
TD2['anomaly'] = Day_End['anomaly']
TD2.head()
```

Out[74]:

	value	day_type	DTC1	anomaly
0	69.880835	0	1	1
1	71.220227	0	1	1
2	70.877805	0	1	1
3	68.959400	0	1	1
4	69.283551	0	1	1

In [75]:
```
a = TD2.loc[TD2['anomaly'] == 1, 'value']
b = TD2.loc[TD2['anomaly'] == -1, 'value']

fig, axs = plt.subplots(figsize=(10, 5))
axs.hist([a,b], bins=32, stacked=True, color=['green', 'blue'])
plt.xlabel('value')
```

```
plt.ylabel('Frequency')
plt.show();
```



EDA(IsolationForest)

In [76]:
```
TD3=TD.copy()
```

In [77]:
```
iso=IsolationForest(n_estimators=50, max_samples='auto', contamination=0.01,max_featur
iso.fit(TD3[['value']])
```

Out[77]:
```
IsolationForest(contamination=0.01, n_estimators=50)
```

In [78]:
```
TD3['scores']=iso.decision_function(TD3[['value']])
TD3['anomaly']=iso.predict(TD3[['value']])
TD3.head()
```

Out[78]:

	value	day_type	DTC1	scores	anomaly
0	69.880835	0	1	0.260968	1
1	71.220227	0	1	0.279137	1
2	70.877805	0	1	0.289274	1
3	68.959400	0	1	0.233767	1
4	69.283551	0	1	0.256446	1

In [79]:
```
TD3[(TD3.value < Ll) | (TD3.value > Ul)].count()
```

Out[79]:
```
value       146
day_type    146
DTC1        146
scores      146
anomaly     146
dtype: int64
```

In [80]:
```
a = TD3.loc[TD3['anomaly'] == 1, 'value']
b = TD3.loc[TD3['anomaly'] == -1, 'value']

fig, axs = plt.subplots(figsize=(10,5))
axs.hist([a,b], bins=32, stacked=True, color=['green', 'blue'])
plt.xlabel('value')
plt.ylabel('Frequency')
plt.show();
```