In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
#import interactive library
from ipywidgets import interact
```

In [4]:
```
#balancing the target variable
from imblearn.over_sampling import RandomOverSampler
```

In [5]:
```
#variable scaling libraries
from sklearn.preprocessing import StandardScaler
```

In [6]:
```
#sklearn_libraries
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
```

In [7]:
```
#feature_selections
from sklearn.feature_selection import f_classif
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
```

In [28]:
```
#evaluation_metrics
from sklearn import metrics
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

#import hyperparameter tuning library
from sklearn.model_selection import GridSearchCV

#other libraries
import math
from collections import Counter
```

In [9]:
```
data=pd.read_csv(r"C:\Users\SENAPATHI REDDY.K\Downloads\waterQuality1.csv")
```

EDA_ANALYSIS

In [10]:
```
data.head()
```

Out[10]:

	aluminium	ammonia	arsenic	barium	cadmium	chloramine	chromium	copper	f
0	1.65	9.08	0.04	2.85	0.007	0.35	0.83	0.17	
1	2.32	21.16	0.01	3.31	0.002	5.28	0.68	0.66	
2	1.01	14.02	0.04	0.58	0.008	4.24	0.53	0.02	
3	1.36	11.33	0.04	2.96	0.001	7.23	0.03	1.66	
4	0.92	24.33	0.03	0.20	0.006	2.67	0.69	0.57	

5 rows × 21 columns

In [11]:
```
data.shape
```

Out[11]:
```
(7999, 21)
```

In [12]:
```
data.columns
```

Out[12]:
```
Index(['aluminium', 'ammonia', 'arsenic', 'barium', 'cadmium', 'chloramine',
       'chromium', 'copper', 'flouride', 'bacteria', 'viruses', 'lead',
       'nitrates', 'nitrites', 'mercury', 'perchlorate', 'radium', 'selenium',
       'silver', 'uranium', 'is_safe'],
      dtype='object')
```

In [13]:
```
data.isnull().sum()
```

Out[13]:
```
aluminium      0
ammonia        0
arsenic        0
barium         0
cadmium        0
chloramine     0
chromium       0
copper         0
flouride       0
bacteria       0
viruses        0
lead           0
nitrates       0
nitrites       0
mercury        0
perchlorate    0
radium         0
selenium       0
silver         0
uranium        0
is_safe        0
dtype: int64
```

In [14]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7999 entries, 0 to 7998
Data columns (total 21 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   aluminium    7999 non-null   float64
 1   ammonia      7999 non-null   object 
 2   arsenic      7999 non-null   float64
 3   barium       7999 non-null   float64
 4   cadmium      7999 non-null   float64
 5   chloramine   7999 non-null   float64
 6   chromium     7999 non-null   float64
 7   copper       7999 non-null   float64
 8   flouride     7999 non-null   float64
 9   bacteria     7999 non-null   float64
 10  viruses      7999 non-null   float64
 11  lead         7999 non-null   float64
 12  nitrates     7999 non-null   float64
 13  nitrites     7999 non-null   float64
 14  mercury      7999 non-null   float64
 15  perchlorate  7999 non-null   float64
 16  radium       7999 non-null   float64
 17  selenium     7999 non-null   float64
 18  silver       7999 non-null   float64
 19  uranium      7999 non-null   float64
 20  is_safe      7999 non-null   object 
```

```
dtypes: float64(19), object(2)
memory usage: 1.3+ MB
```

In [15]:
```
#drop missing values
data['ammonia'].value_counts()['#NUM!']
data = data[data['ammonia'].str.contains('#NUM!') == False]
```

In [16]:
```
#convert both columns to numeric data type
data['ammonia'] = pd.to_numeric(data['ammonia'])
data['is_safe'] = pd.to_numeric(data['is_safe'])
```

In [17]:
```
#statistic metrics for continuous variables without scientific notation
data.describe().apply(lambda s: s.apply(lambda x: format(x, 'f')))
```

Out[17]:

	aluminium	ammonia	arsenic	barium	cadmium	chloramine	7
count	7996.000000	7996.000000	7996.000000	7996.000000	7996.000000	7996.000000	
mean	0.666396	14.278212	0.161477	1.567928	0.042803	2.177589	
std	1.265323	8.878930	0.252632	1.216227	0.036049	2.567210	
min	0.000000	-0.080000	0.000000	0.000000	0.000000	0.000000	
25%	0.040000	6.577500	0.030000	0.560000	0.008000	0.100000	
50%	0.070000	14.130000	0.050000	1.190000	0.040000	0.530000	
75%	0.280000	22.132500	0.100000	2.482500	0.070000	4.240000	
max	5.050000	29.840000	1.050000	4.940000	0.130000	8.680000	

8 rows × 21 columns

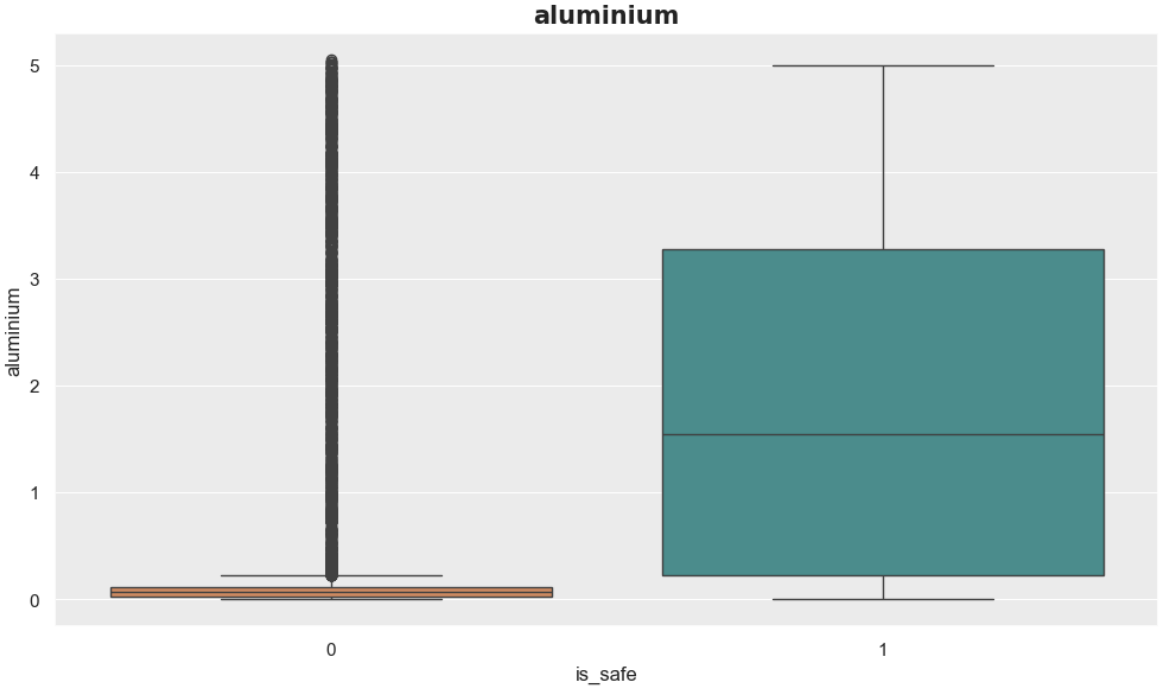
In [18]:
```
#create a list of numerical features and plot them
list_of_num_features = data.loc[:, data.columns != 'is_safe']
palette_features = ['#E68753', '#409996']
sns.set(rc={'axes.facecolor':'#ECECEC'}) #background color of all plots

for feature in list_of_num_features:
```

```
    plt.figure(figsize=(12,6.5))
    plt.title(feature, fontsize=15, fontweight='bold', fontname='Helvetica', ha=
    ax = sns.boxplot(
    x='is_safe',
    y=list_of_num_features[feature],
    data=data,
    hue='is_safe',
    palette=palette_features,
    legend=False
)
```

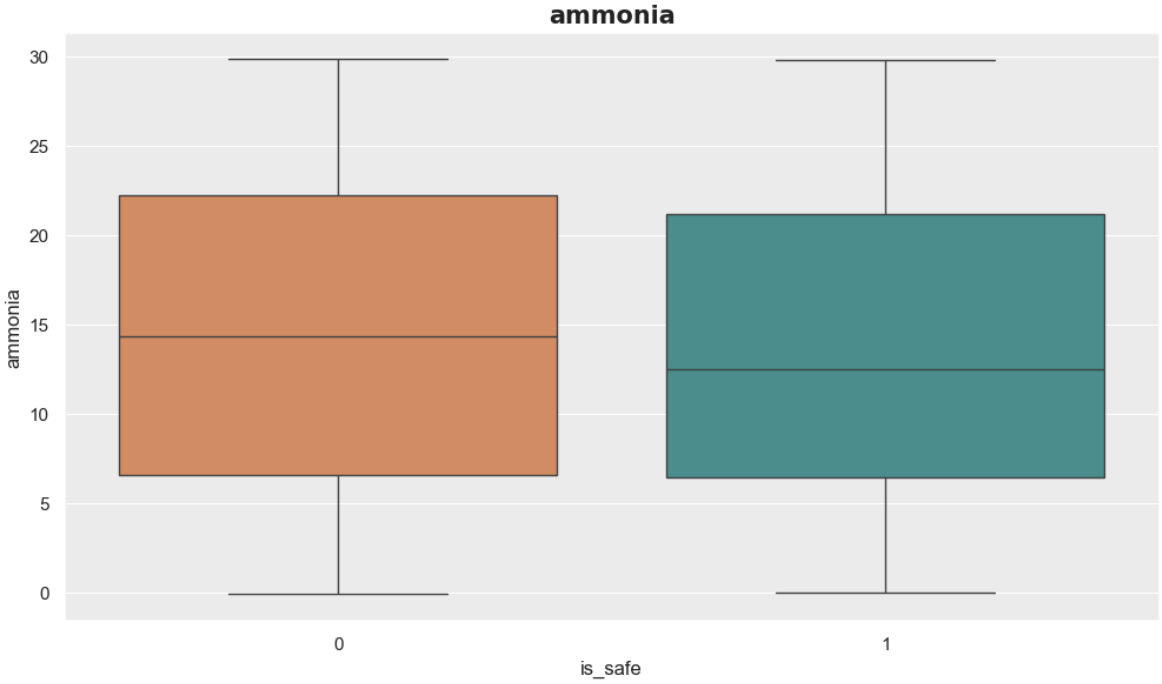
```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```

```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```

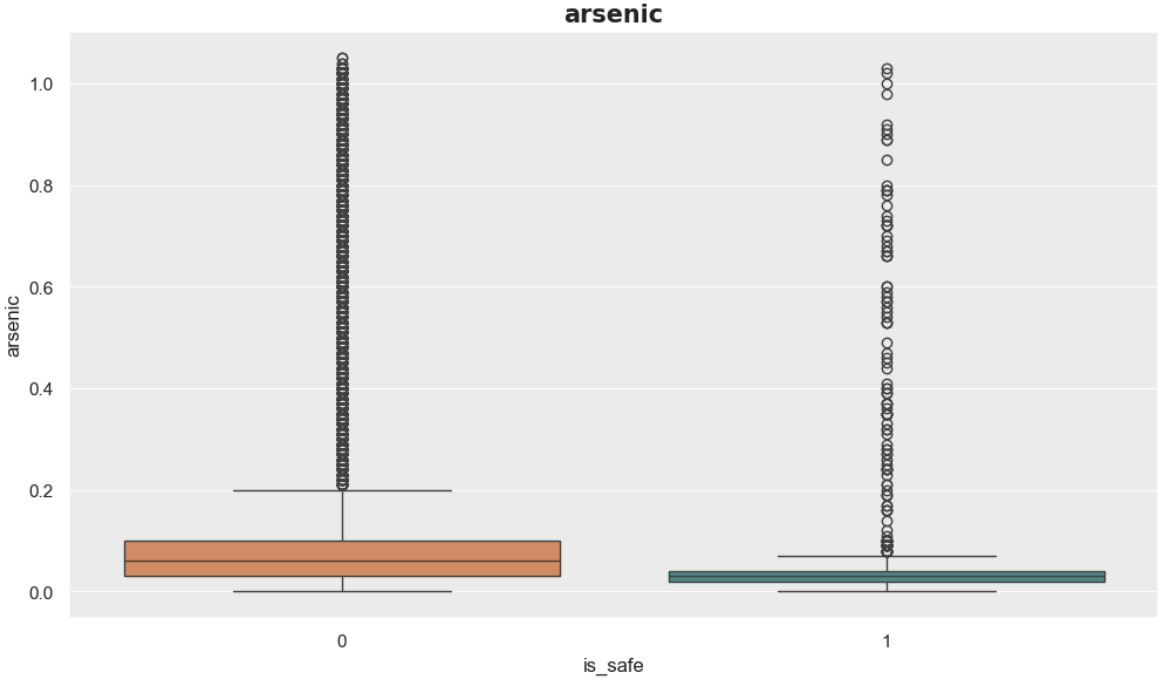
```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```



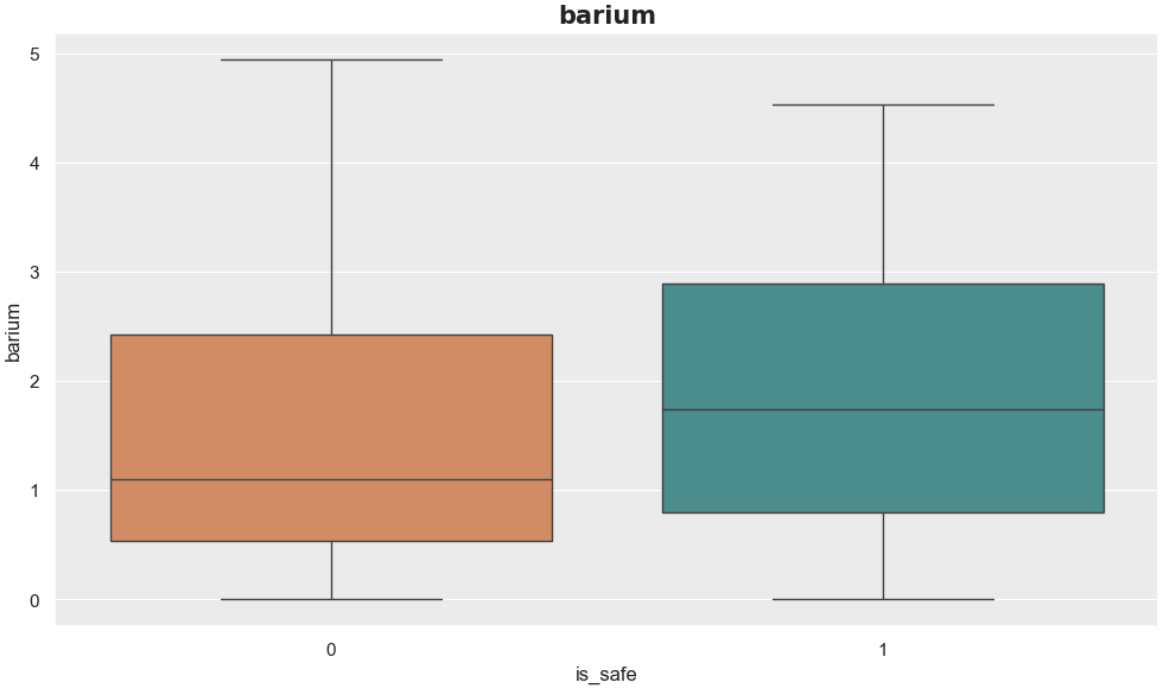

```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```



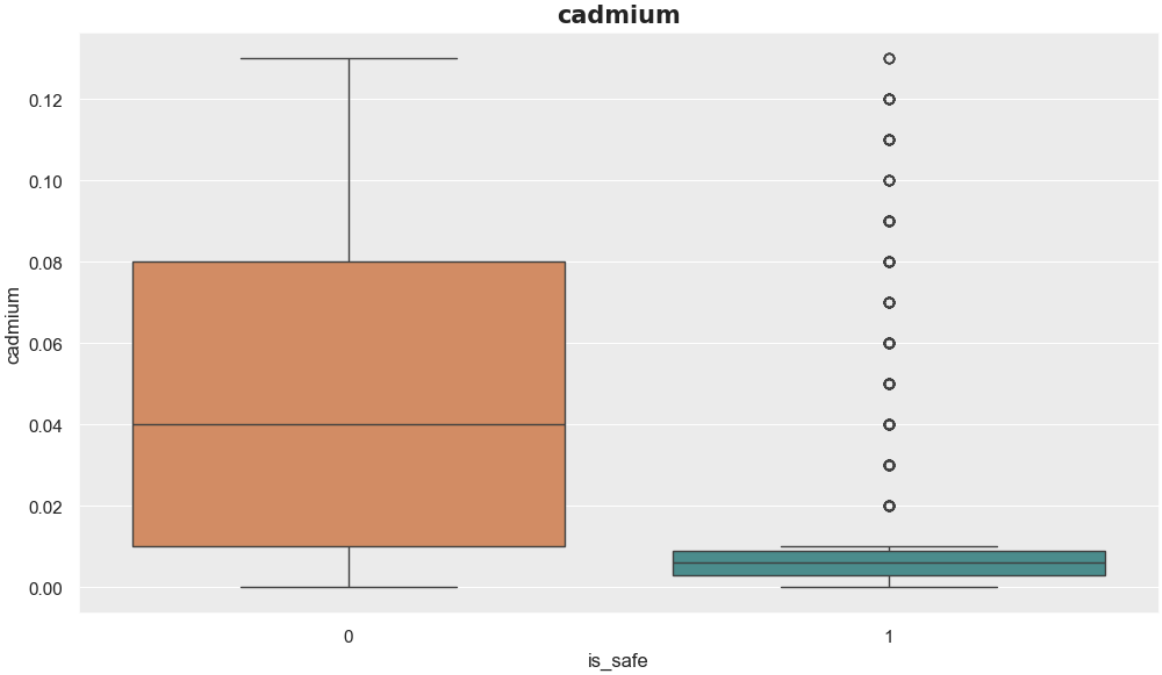

```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```



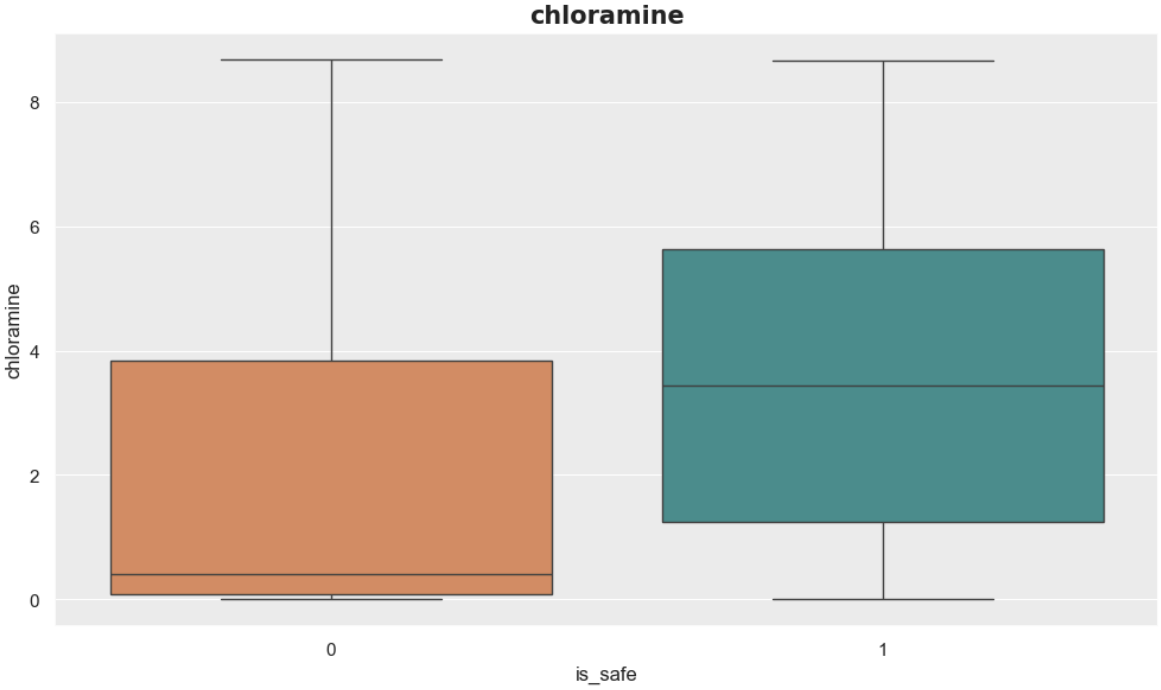

```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```



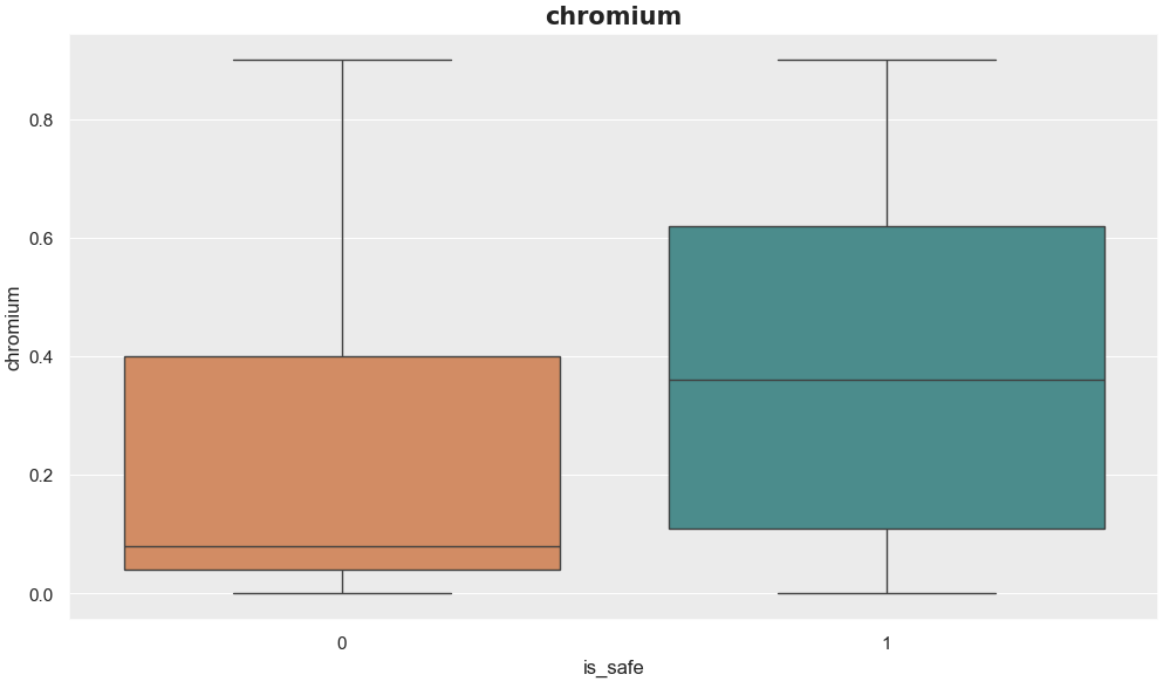

```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```



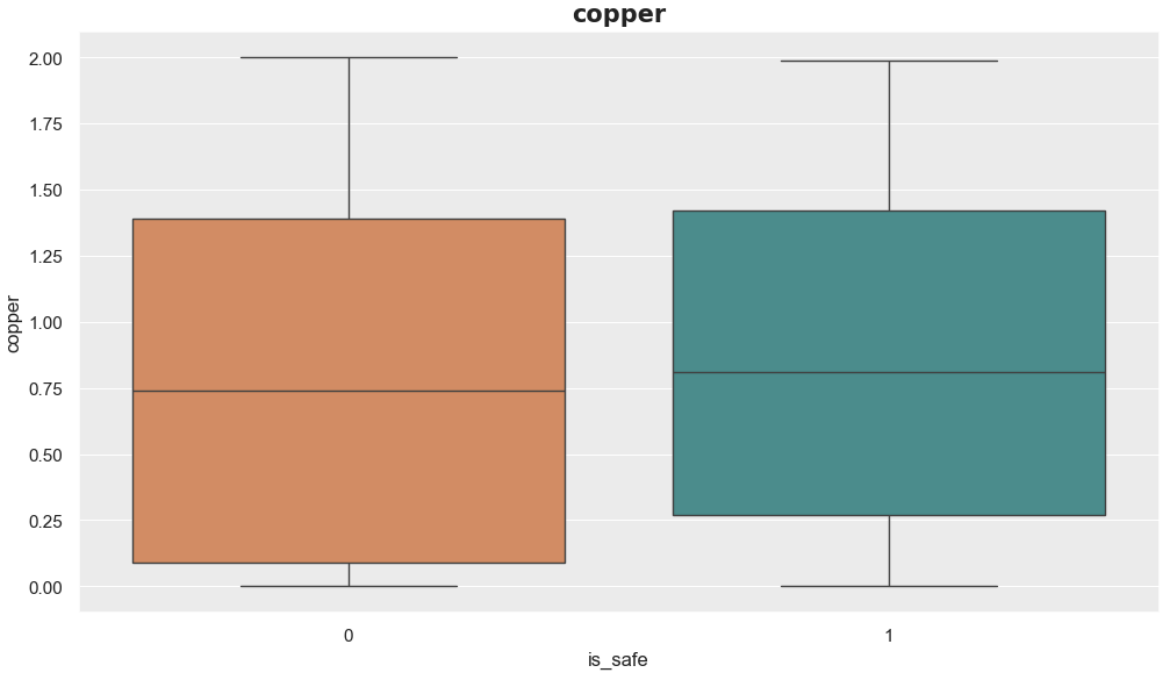

```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```



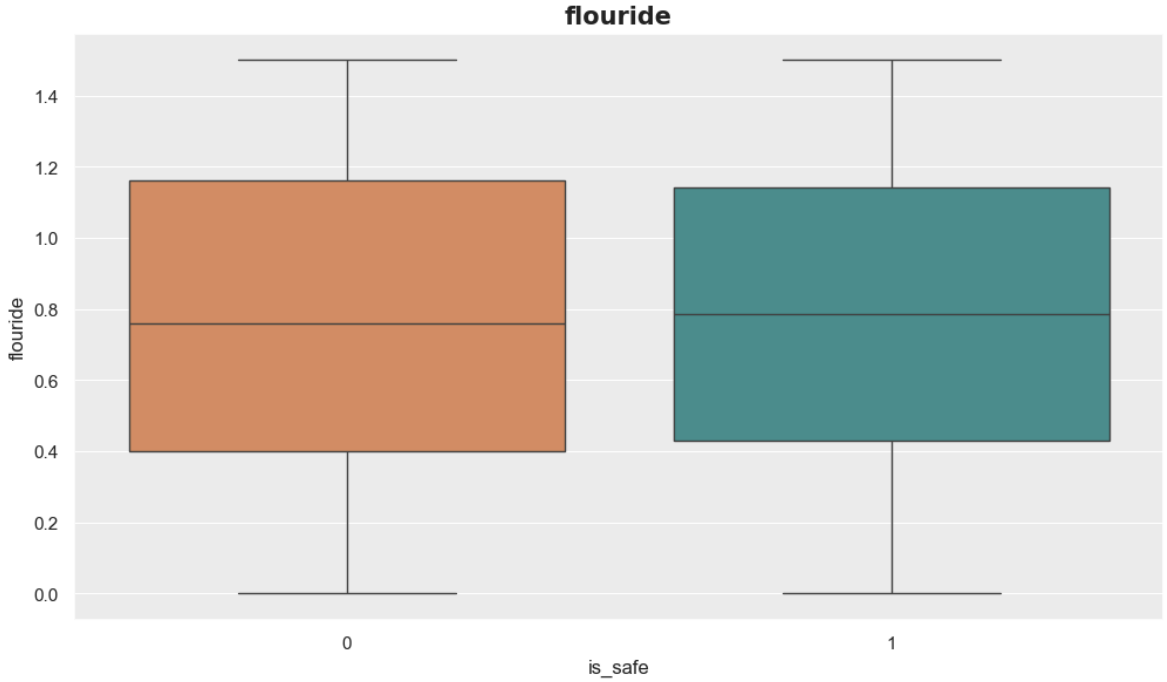

```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```



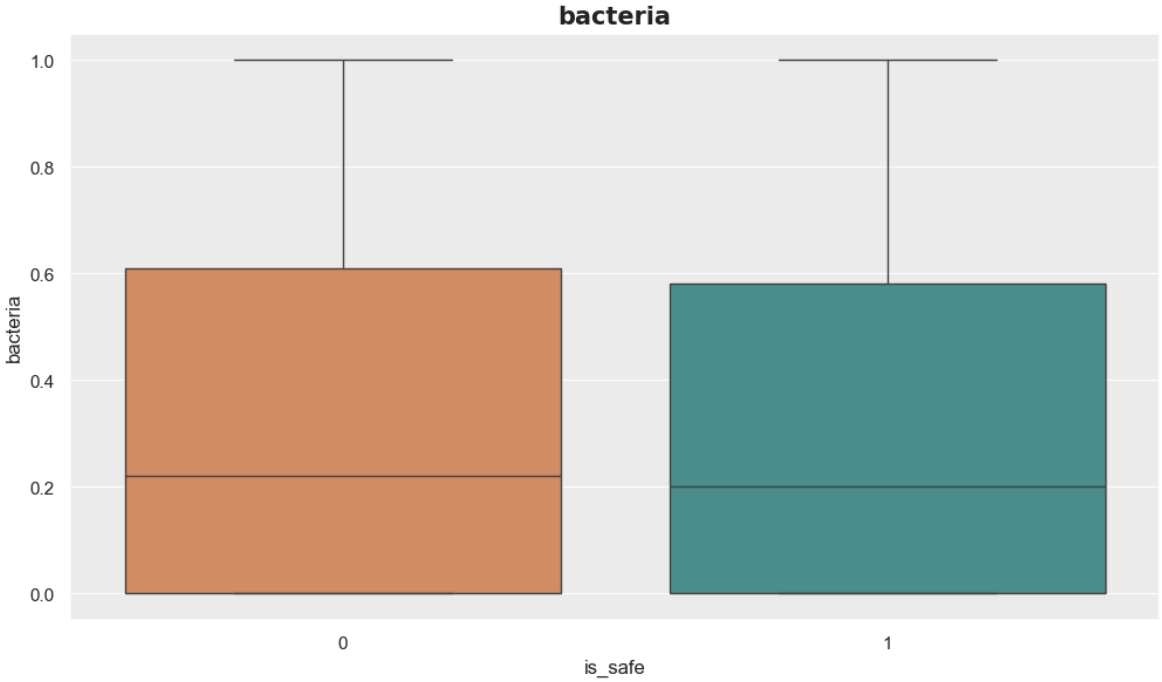

```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```



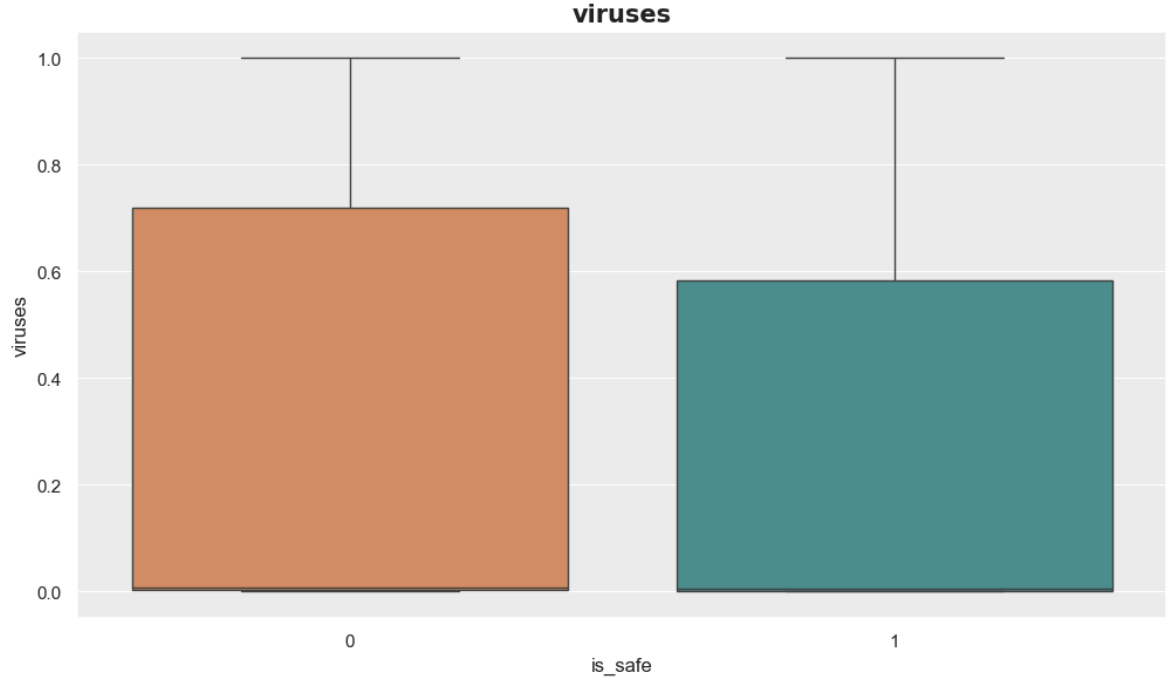

```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```



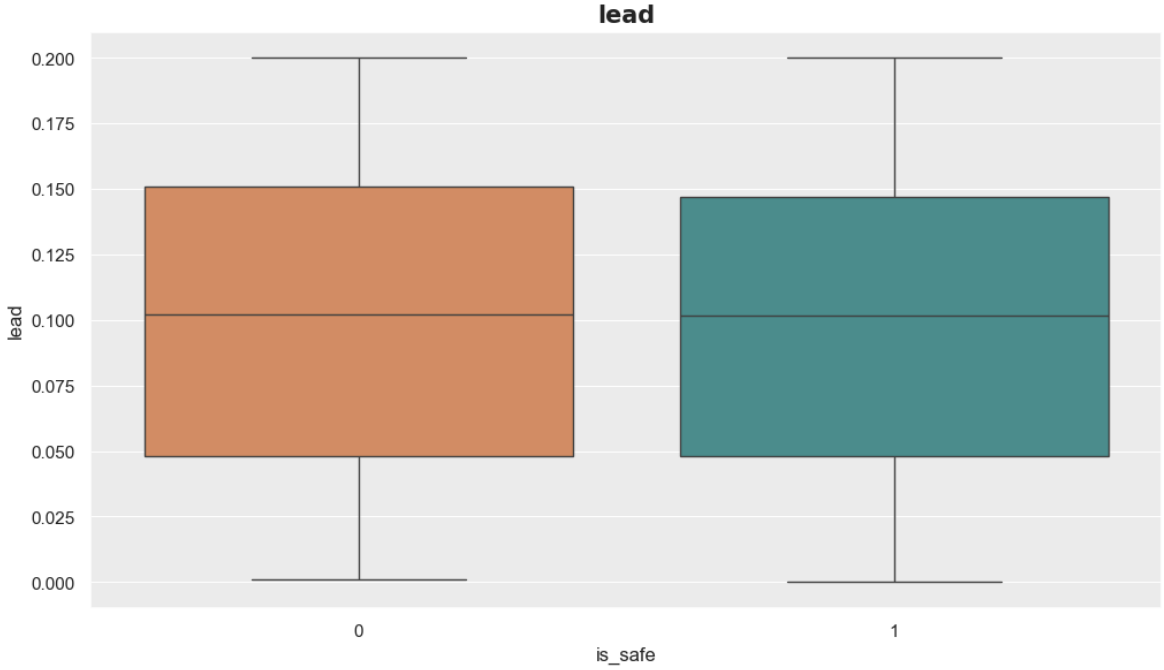

```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```



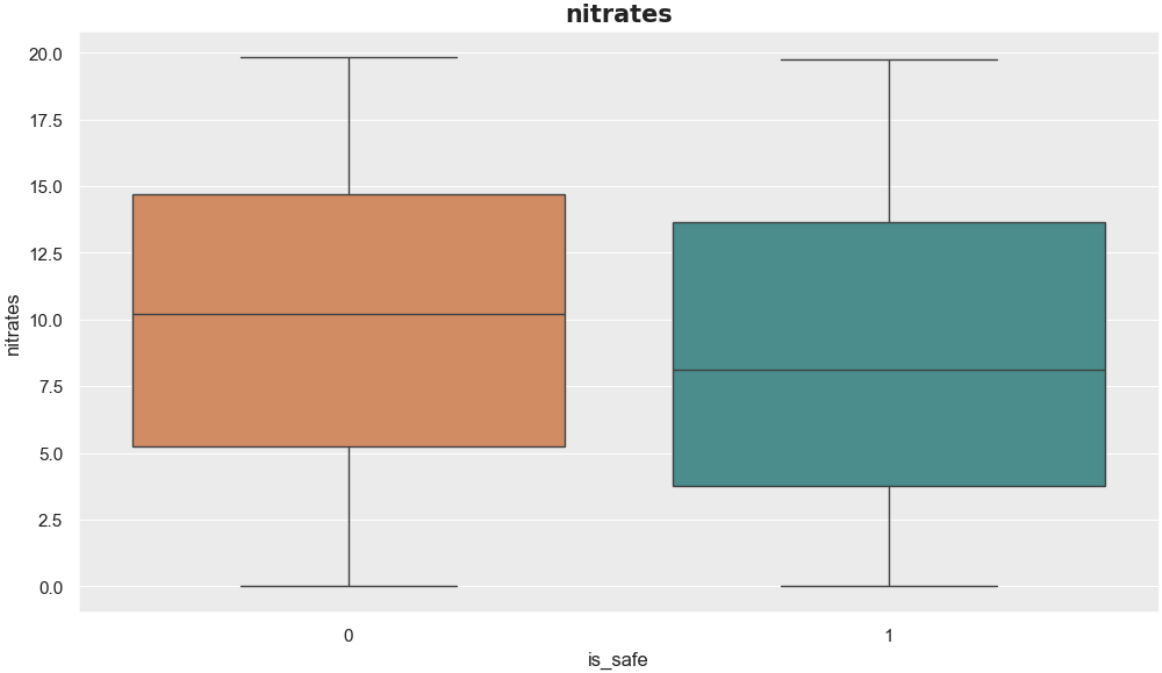

```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```



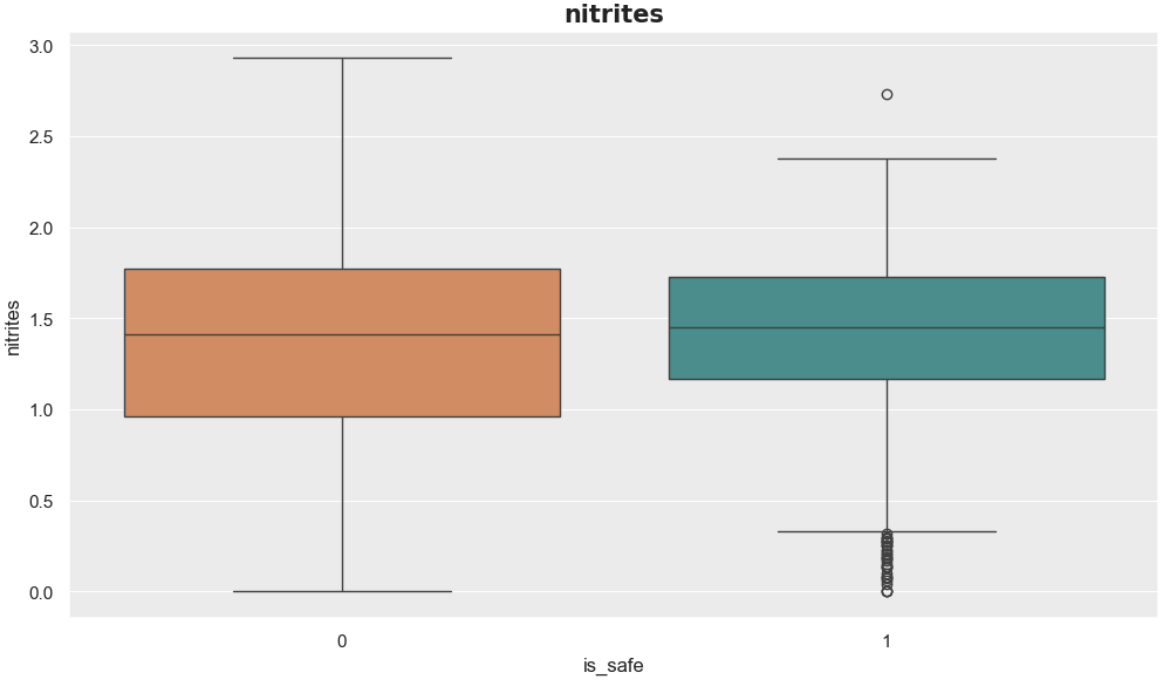

```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```



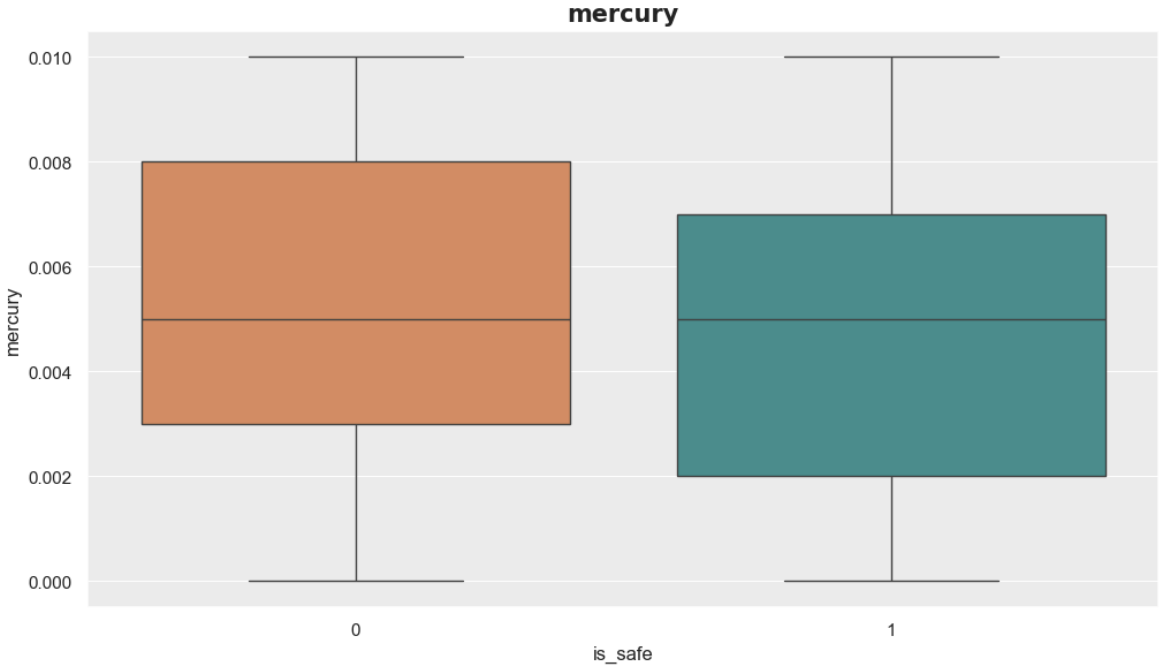

```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```



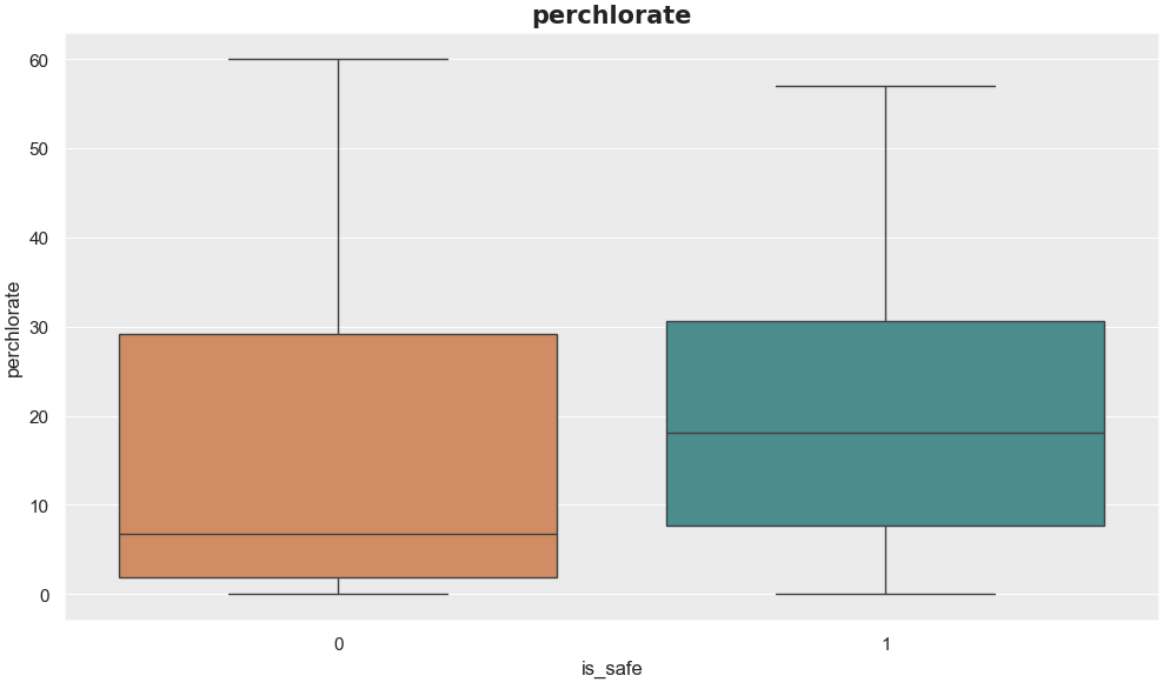

```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```



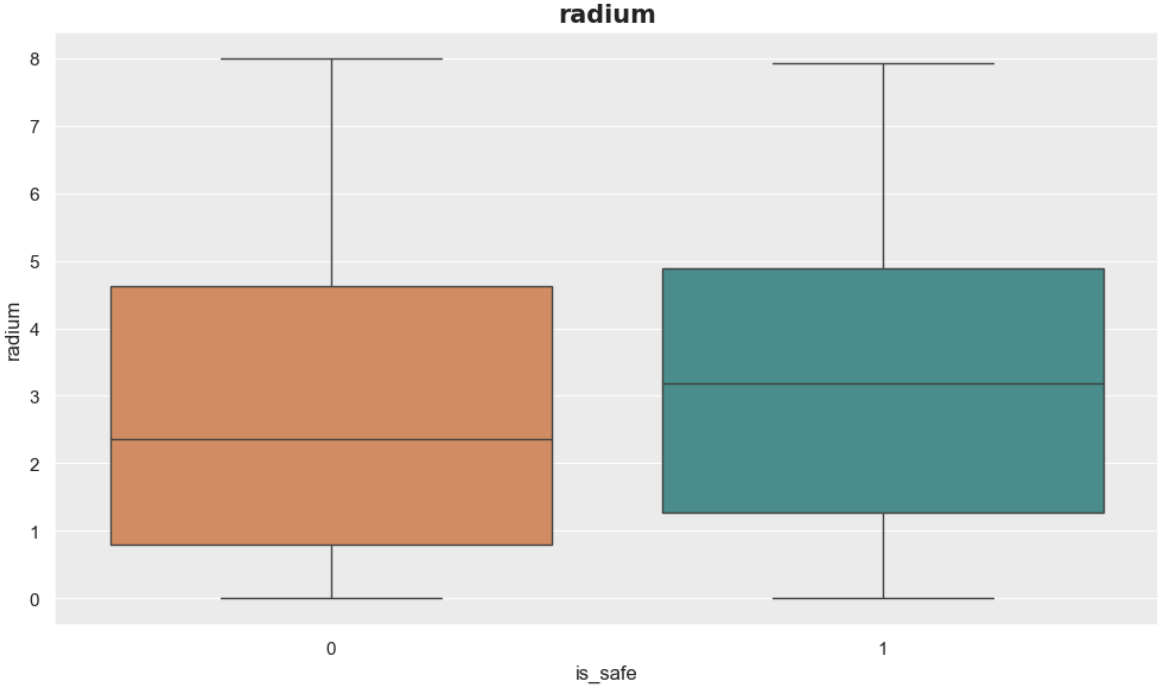

```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```



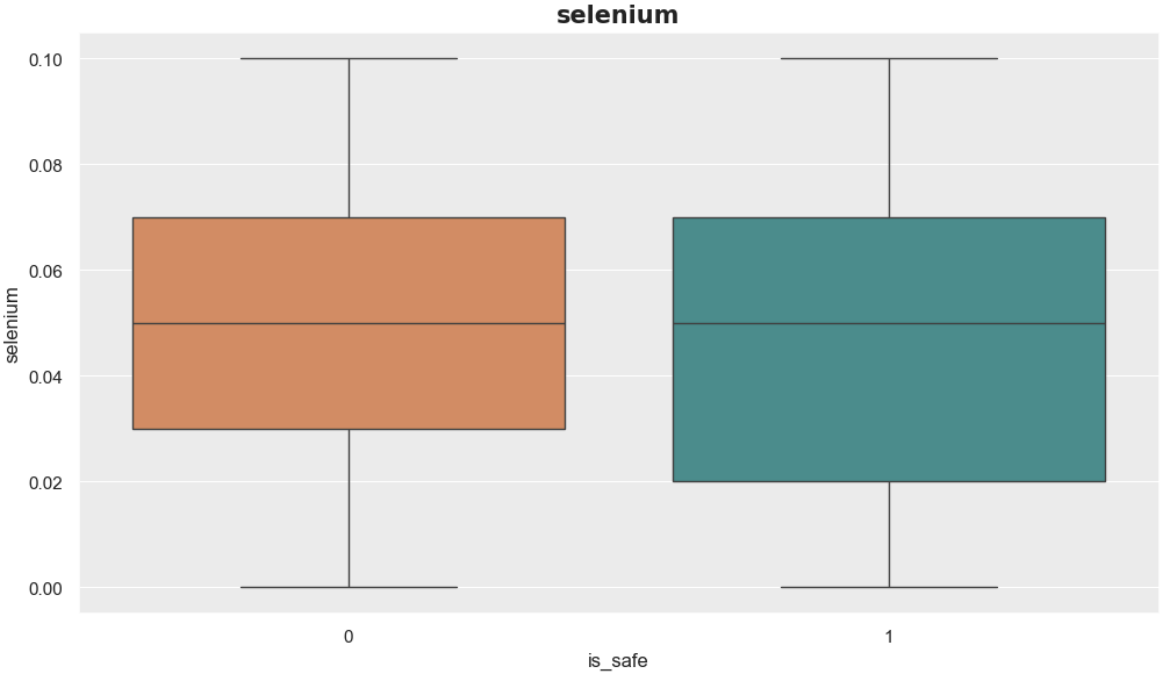

```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```



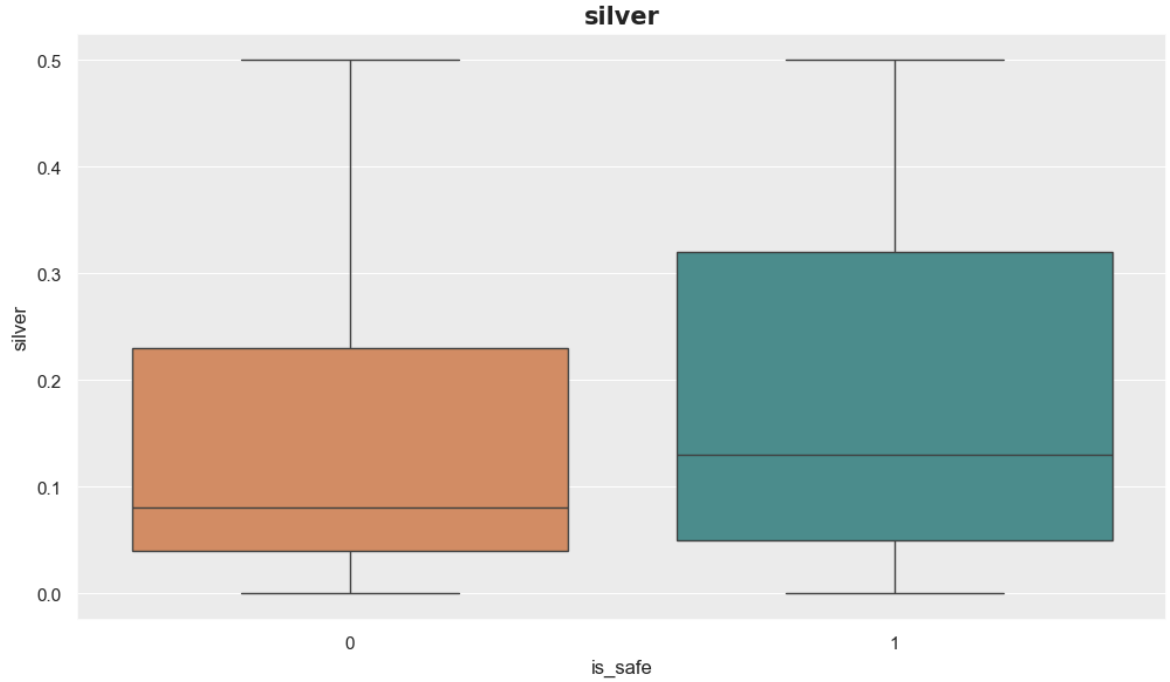

```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```



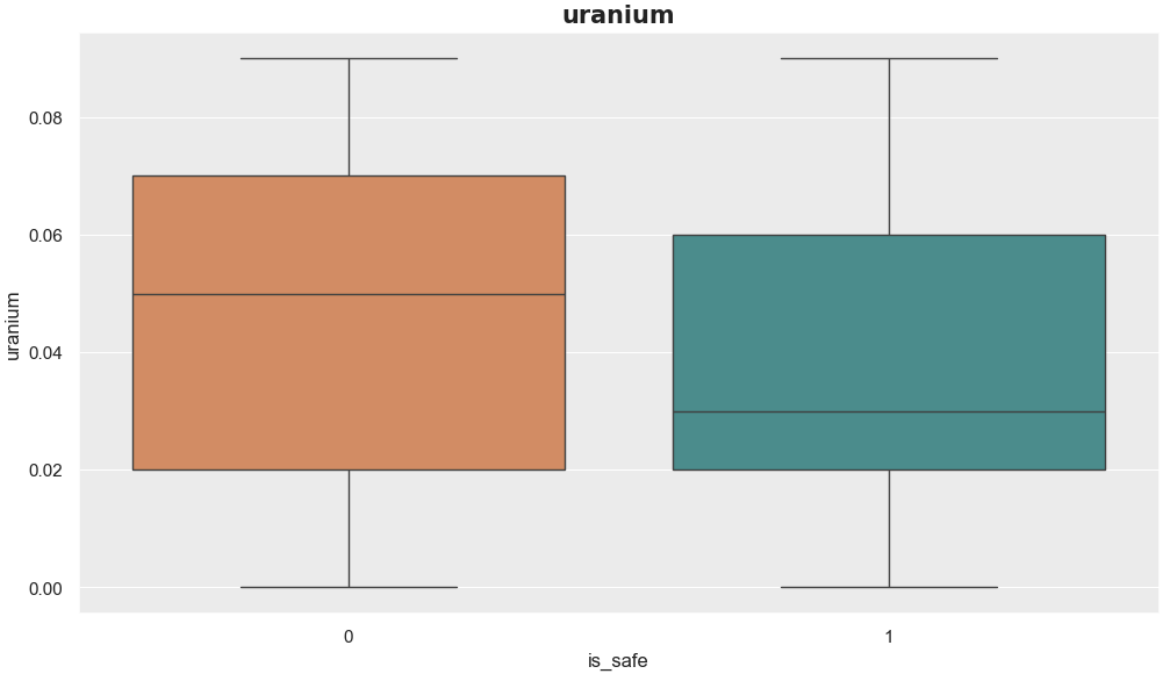

```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```




```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```




```
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```



In [20]:
```
#balance the target variable
columns = data.columns
columns = [c for c in columns if c not in ['is_safe']]
y = data['is_safe'] #prior target variable
x = data[columns]  #prior features
```

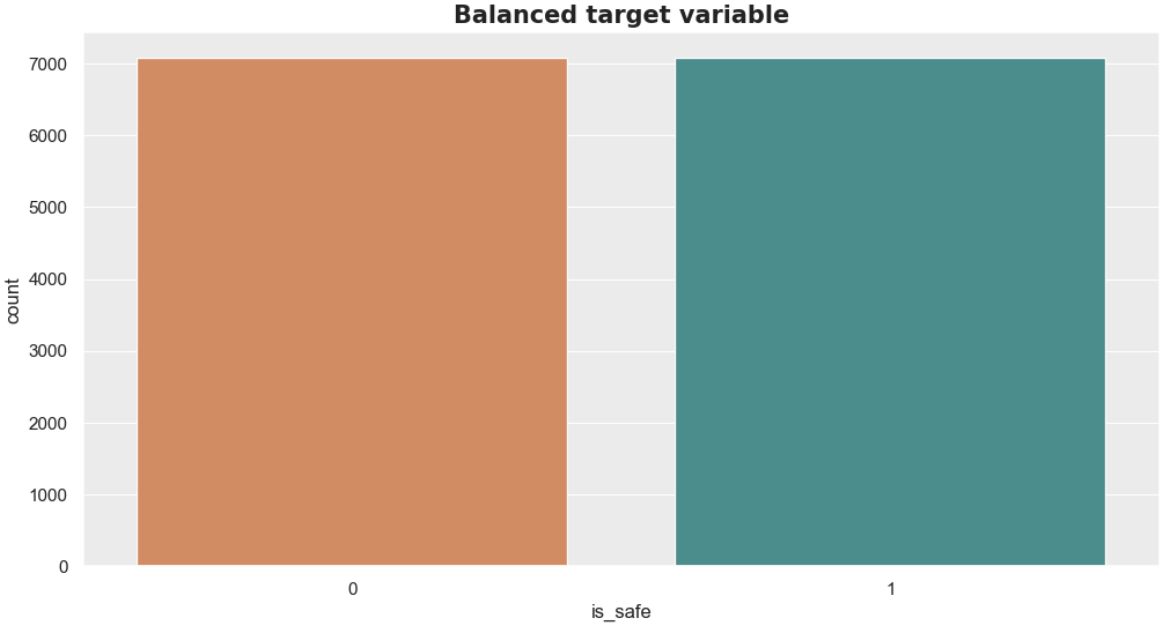
```
ros = RandomOverSampler(sampling_strategy='minority')
x, y = ros.fit_resample(x, y) #y_train as balanced target variable
print(f"Imbalanced target class: {(y)}\n\nBalanced target class: {Counter(y)}\n"
print(x.shape[0] - data.shape[0], 'new random picked points')
```

```
Imbalanced target class: 0        1
1         1
2         0
3         1
4         1
        ..
14163     1
14164     1
14165     1
14166     1
14167     1
Name: is_safe, Length: 14168, dtype: int64

Balanced target class: Counter({1: 7084, 0: 7084})

6172 new random picked points
```

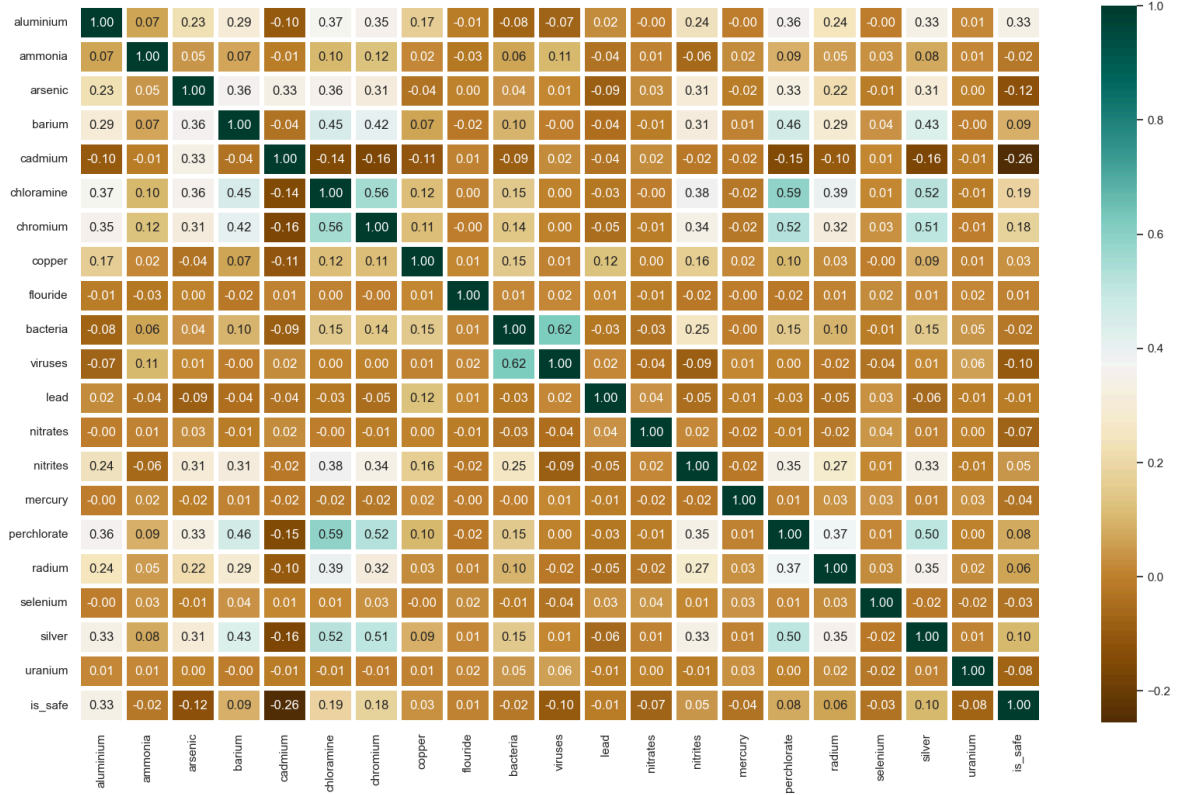
In [21]:
```
#plot the balanced target variable
sns.set(rc={'axes.facecolor':'#ECECEC'}) #background color of plot
plt.figure(figsize=(12,6))
plt.title("Balanced target variable", fontsize=15, fontweight='bold', fontname='
```

```
ax = sns.countplot(x=y, data=data, palette=palette_features)
plt.show()
```

```
C:\Users\SENAPATHI REDDY.K\AppData\Local\Temp\ipykernel_18828\1791969356.py:5: Fu
tureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  ax = sns.countplot(x=y, data=data, palette=palette_features)
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
findfont: Font family 'Helvetica' not found.
```



In [22]:
```
# plotting correlation matrix to notice relationships or lack of it between vari
corr = data.corr()

plt.figure(figsize = (20, 12))
sns.heatmap(corr, xticklabels = corr.columns, yticklabels = corr.columns, linewi
plt.show()
```


aluminium 1.00 0.07 0.23 0.29 -0.10 0.37 0.35 0.17 -0.01 -0.08 -0.07 0.02 -0.00 0.24 -0.00 0.36 0.24 -0.00 0.33 0.01 0.33
ammonia 0.07 1.00 0.05 0.07 -0.01 0.10 0.12 0.02 -0.03 0.06 0.11 -0.04 0.01 -0.06 0.02 0.09 0.05 0.03 0.08 0.01 -0.02
arsenic 0.23 0.05 1.00 0.36 0.33 0.36 0.31 -0.04 0.00 0.04 0.01 -0.09 0.03 0.31 -0.02 0.33 0.22 -0.01 0.31 0.00 -0.12
barium 0.29 0.07 0.36 1.00 -0.04 0.45 0.42 0.07 -0.02 0.10 -0.00 -0.04 -0.01 0.31 0.01 0.46 0.29 0.04 0.43 -0.00 0.09
cadmium -0.10 -0.01 0.33 -0.04 1.00 -0.14 -0.16 -0.11 0.01 -0.09 0.02 -0.04 0.02 -0.02 -0.02 -0.15 -0.10 0.01 -0.16 -0.01 -0.26
chloramine 0.37 0.10 0.36 0.45 -0.14 1.00 0.56 0.12 0.00 0.15 0.00 -0.03 -0.00 0.38 -0.02 0.59 0.39 0.01 0.52 -0.01 0.19
chromium 0.35 0.12 0.31 0.42 -0.16 0.56 1.00 0.11 -0.00 0.14 0.00 -0.05 -0.01 0.34 -0.02 0.52 0.32 0.03 0.51 -0.01 0.18
copper 0.17 0.02 -0.04 0.07 -0.11 0.12 0.11 1.00 0.01 0.15 0.01 0.12 0.00 0.16 0.02 0.10 0.03 -0.00 0.09 0.01 0.03
flouride -0.01 -0.03 0.00 -0.02 0.01 0.00 -0.00 0.01 1.00 0.01 0.02 0.01 -0.01 -0.02 -0.00 -0.02 0.01 0.02 0.01 0.02 0.01
bacteria -0.08 0.06 0.04 0.10 -0.09 0.15 0.14 0.15 0.01 1.00 0.62 -0.03 -0.03 0.25 -0.00 0.15 0.10 -0.01 0.15 0.05 -0.02
viruses -0.07 0.11 0.01 -0.00 0.02 0.00 0.00 0.01 0.02 0.62 1.00 0.02 -0.04 -0.09 0.01 0.00 -0.02 -0.04 0.01 0.06 -0.10
lead 0.02 -0.04 -0.09 -0.04 -0.04 -0.03 -0.05 0.12 0.01 -0.03 0.02 1.00 0.04 -0.05 -0.01 -0.03 -0.05 0.03 -0.06 -0.01 -0.01
nitrates -0.00 0.01 0.03 -0.01 0.02 -0.00 -0.01 0.00 -0.01 -0.03 -0.04 0.04 1.00 0.02 -0.02 -0.01 -0.02 0.04 0.01 0.00 -0.07
nitrites 0.24 -0.06 0.31 0.31 -0.02 0.38 0.34 0.16 -0.02 0.25 -0.09 -0.05 0.02 1.00 -0.02 0.35 0.27 0.01 0.33 -0.01 0.05
mercury -0.00 0.02 -0.02 0.01 -0.02 -0.02 -0.02 0.02 -0.00 -0.00 0.01 -0.01 -0.02 -0.02 1.00 0.01 0.03 0.03 0.01 0.03 -0.04
perchlorate 0.36 0.09 0.33 0.46 -0.15 0.59 0.52 0.10 -0.02 0.15 0.00 -0.03 -0.01 0.35 0.01 1.00 0.37 0.01 0.50 0.00 0.08
radium 0.24 0.05 0.22 0.29 -0.10 0.39 0.32 0.03 0.01 0.10 -0.02 -0.05 -0.02 0.27 0.03 0.37 1.00 0.03 0.35 0.02 0.06
selenium -0.00 0.03 -0.01 0.04 0.01 0.01 0.03 -0.00 0.02 -0.01 -0.04 0.03 0.04 0.01 0.03 0.01 0.03 1.00 -0.02 -0.02 -0.03
silver 0.33 0.08 0.31 0.43 -0.16 0.52 0.51 0.09 0.01 0.15 0.01 -0.06 0.01 0.33 0.01 0.50 0.35 -0.02 1.00 0.01 0.10
uranium 0.01 0.01 0.00 -0.00 -0.01 -0.01 -0.01 0.01 0.02 0.05 0.06 -0.01 0.00 -0.01 0.03 0.00 0.02 -0.02 0.01 1.00 -0.08
is_safe 0.33 -0.02 -0.12 0.09 -0.26 0.19 0.18 0.03 0.01 -0.02 -0.10 -0.01 -0.07 0.05 -0.04 0.08 0.06 -0.03 0.10 -0.08 1.00
aluminium ammonia arsenic barium cadmium chloramine chromium copper flouride bacteria viruses lead nitrates nitrites mercury perchlorate radium selenium silver uranium is_safe
1.0 0.8 0.6 0.4 0.2 0.0 −0.2

In [23]:
```
#preparation of training and testing_data
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42
```

In [24]:
```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

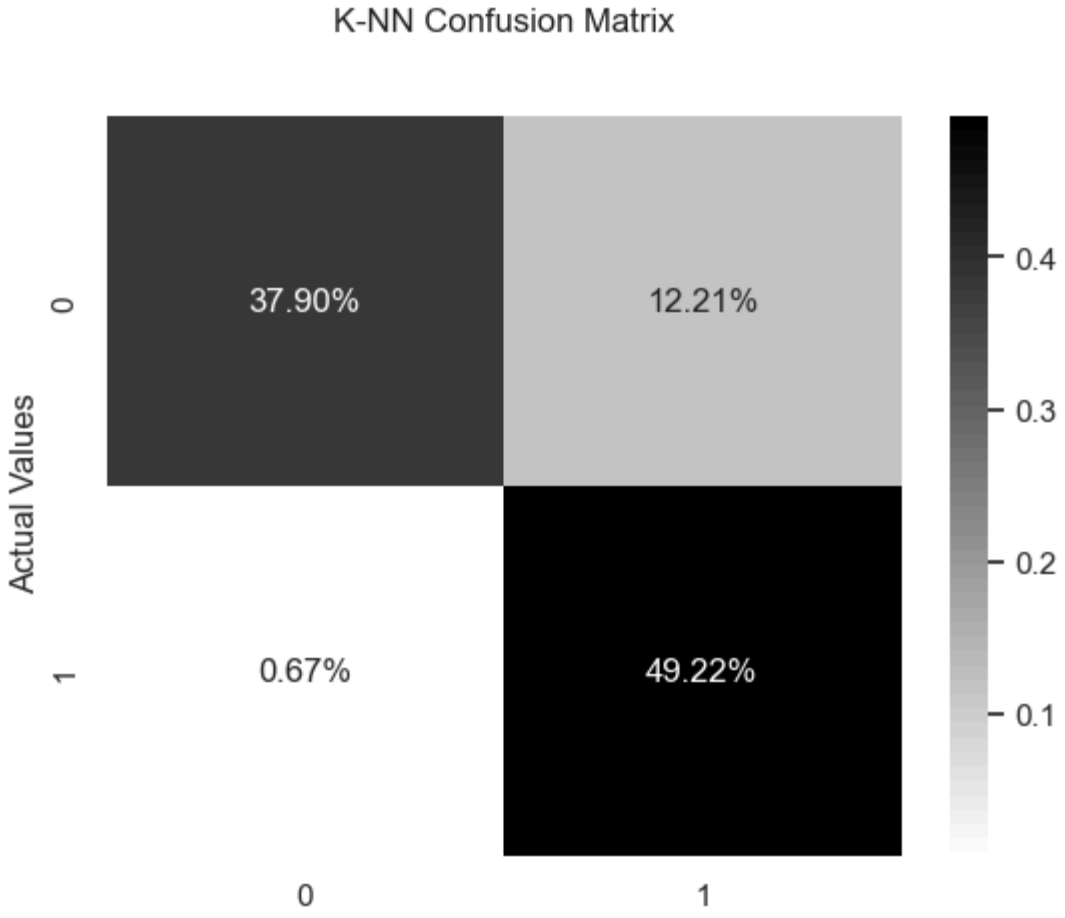
Out[24]: ((11334, 20), (2834, 20), (11334,), (2834,))

In [26]:
```
knnc=KNeighborsClassifier()
knnc.fit(x_train,y_train)
y_pred=knnc.predict(x_test)
```

In [29]:
```
cf_matrix=confusion_matrix(y_test,y_pred)
print(cf_matrix)
```

```
[[1074  346]
 [  19 1395]]
```

In [31]:
```
ax = sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True, fmt='.2%', cmap='binar
ax.set_title('K-NN Confusion Matrix\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');
ax.xaxis.set_ticklabels(['0','1'])
ax.yaxis.set_ticklabels(['0','1'])
plt.show()
```


K-NN Confusion Matrix
37.90%
12.21%
0.67%
49.22%
Actual Values
0
1
0
1
0.4
0.3
0.2
0.1

Predicted Values

In [32]:
```
print(classification_report(y_test,y_pred))
print("accuracy_score",accuracy_score(y_test,y_pred))
print("precision_score",precision_score(y_test,y_pred))
print("f1-score",f1_score(y_test,y_pred))
print("recall_score",recall_score(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.76      0.85      1420
           1       0.80      0.99      0.88      1414

    accuracy                           0.87      2834
   macro avg       0.89      0.87      0.87      2834
weighted avg       0.89      0.87      0.87      2834

accuracy_score 0.8712067748764997
precision_score 0.8012636415852958
f1-score 0.884310618066561
recall_score 0.9865629420084866
```

In [33]:
```
x_train,x_test,y_train,y_test = train_test_split(x, y,test_size=0.3, random_stat
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[33]: ((9917, 20), (4251, 20), (9917,), (4251,))

In [38]:
```
sc = StandardScaler()
sc.fit(x_train)
x_train_std = sc.transform(x_train)
x_test_std = sc.transform(x_test)
print('After standardizing our features, the first 5 rows of our data now look l
print(pd.DataFrame(x_train_std).head())
```

After standardizing our features, the first 5 rows of our data now look like thi
s:

	0	1	2	3	4	5	6	\
0	-0.747105	-1.287465	-0.515795	-1.229938	-0.879215	0.502308	0.900971	
1	0.470680	-1.600619	-0.426214	1.078939	-0.681569	1.303799	0.226259	
2	-0.148086	0.112650	-0.471004	-0.722313	-0.794510	-0.452578	0.510348	
3	1.168438	0.131938	-0.515795	0.563126	-0.681569	-0.425734	-0.199875	
4	-0.286321	-0.534081	-0.515795	1.463752	-0.879215	0.180178	-0.839076	

	7	8	9	10	11	12	13	\
0	-0.129825	1.104969	1.711217	1.587798	-0.037843	-1.600573	-0.134877	
1	0.429102	-0.396231	1.281916	1.207030	1.370857	0.496324	0.956896	
2	0.972504	-0.765758	-0.956581	-0.778405	1.701293	1.043028	1.033512	
3	0.444628	0.850920	-0.956581	-0.778405	0.379550	-1.233706	0.037508	
4	0.040958	1.174255	0.239328	-0.764807	-1.446543	0.361447	0.056662	

	14	15	16	17	18	19
0	0.330781	0.496439	-0.905075	0.728547	-0.025737	1.425689
1	0.994352	1.101963	0.319034	0.382140	1.116711	1.052404
2	-1.328144	-0.611256	0.766137	0.035734	-0.832171	0.679118
3	-0.664574	-0.173528	1.933815	1.767766	-1.033780	-0.440738
4	0.994352	0.743269	1.530119	-1.349892	1.788740	-0.067453

In [39]:
```
# find best scored 10 features
select_feature = SelectKBest(f_classif, k=10).fit(x_train, y_train)
```

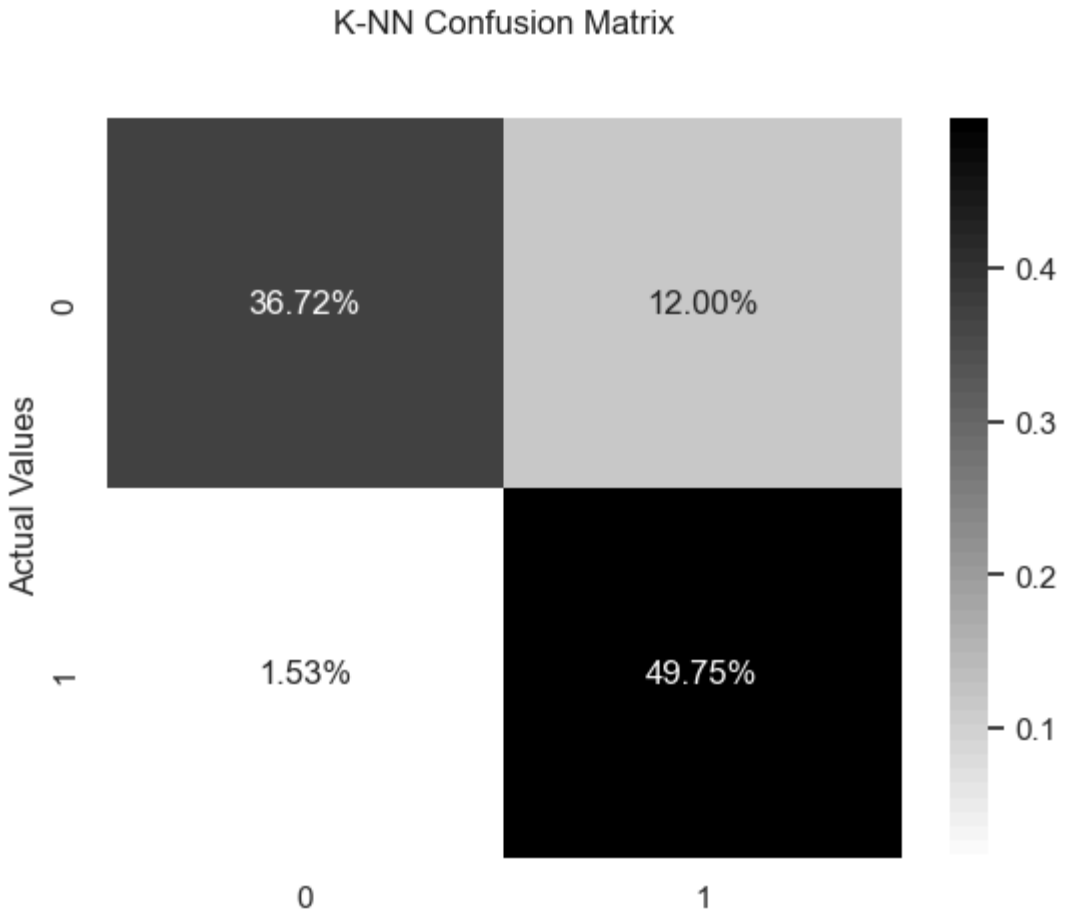
```
print('Score list:', select_feature.scores_)
print('Feature list:', x_train.columns)
```

```
Score list: [2.20931713e+03 6.18990165e+00 4.91501221e+02 1.81201115e+02
 2.06393780e+03 9.72225517e+02 7.59077165e+02 1.49932412e+01
 1.26787348e+00 1.17728614e+01 2.47876336e+02 3.85561481e+00
 1.11624492e+02 5.19701896e+01 4.76808986e+01 1.56665064e+02
 9.93507786e+01 1.91311693e+01 2.19988180e+02 1.57347696e+02]
Feature list: Index(['aluminium', 'ammonia', 'arsenic', 'barium', 'cadmium', 'chl
oramine',
       'chromium', 'copper', 'flouride', 'bacteria', 'viruses', 'lead',
       'nitrates', 'nitrites', 'mercury', 'perchlorate', 'radium', 'selenium',
       'silver', 'uranium'],
      dtype='object')
```

In [40]:
```
x_train_2 = select_feature.transform(x_train)
x_test_2 = select_feature.transform(x_test)

knnc = KNeighborsClassifier()
knnc = knnc.fit(x_train_2,y_train)

y_pred_knnc = knnc.predict(x_test_2)

cf_matrix_knnc = confusion_matrix(y_test, y_pred_knnc)
print(cf_matrix_knnc)

ax = sns.heatmap(cf_matrix_knnc/np.sum(cf_matrix_knnc), annot=True, fmt='.2%', c
```

```
ax.set_title('K-NN Confusion Matrix\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['0','1'])
ax.yaxis.set_ticklabels(['0','1'])

plt.show()
```

```
[[1561  510]
 [  65 2115]]
```


K-NN Confusion Matrix
36.72%
12.00%
1.53%
49.75%
Actual Values
0
1
0
1
0.4
0.3
0.2
0.1

Predicted Values

In [41]:
```
#evaluation of K-NN: metrics pivot chart
print(classification_report(y_test, y_pred_knnc))

print('Accuracy Score : ' + str(round(accuracy_score(y_test,y_pred_knnc),3)))
print('Precision Score : ' + str(round(precision_score(y_test,y_pred_knnc,pos_la
print('Recall Score : ' + str(round(recall_score(y_test,y_pred_knnc,pos_label=0)
print('F-Score : ' + str(round(f1_score(y_test,y_pred_knnc,pos_label=0),3)))
```

```
              precision    recall  f1-score   support

           0       0.96      0.75      0.84      2071
           1       0.81      0.97      0.88      2180

    accuracy                           0.86      4251
   macro avg       0.88      0.86      0.86      4251
weighted avg       0.88      0.86      0.86      4251

Accuracy Score : 0.865
Precision Score : 0.96
Recall Score : 0.754
F-Score : 0.844
```

In [42]:
```
param_grid = {'n_neighbors':np.arange(1,40), 'metric':['euclidean', 'manhattan',
knn = KNeighborsClassifier()
knn_cv = GridSearchCV(knn,param_grid,cv=5)
knn_cv.fit(x_train_2,y_train)

y_pred_knn_cv = knn_cv.predict(X_test_2)

print("Best Score:" + str(knn_cv.best_score_))
print("Best Parameters: " + str(knn_cv.best_params_))

cf_matrix_knn_cv = confusion_matrix(y_test, y_pred_knn_cv)
print(cf_matrix_knn_cv)

ax = sns.heatmap(cf_matrix_knn_cv/np.sum(cf_matrix_knn_cv), annot=True, fmt='.2%
ax.set_title('K-NN Confusion Matrix\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['0','1'])
ax.yaxis.set_ticklabels(['0','1'])


plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[42], line 6
      3 knn_cv = GridSearchCV(knn,param_grid,cv=5)
      4 knn_cv.fit(x_train_2,y_train)
----> 6 y_pred_knn_cv = knn_cv.predict(X_test_2)
      8 print("Best Score:" + str(knn_cv.best_score_))
      9 print("Best Parameters: " + str(knn_cv.best_params_))

NameError: name 'X_test_2' is not defined
```

In [ ]: