Predict occurrence of kidney stones using Classification Algorithms

We eventually want to predict some target based on numerous inputs . In this situation, we wish to predict the presence of Kidney Stones, where '1 = we observe a presence' and '0 = we detect no presence.' As a result, this is a 'Binary Classification' issue in which something exists or it does not.

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

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

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

Out[3]:

	id	gravity	ph	osmo	cond	urea	calc	target
0	0	1.013	6.19	443	14.8	124	1.45	0
1	1	1.025	5.40	703	23.6	394	4.18	0
2	2	1.009	6.13	371	24.5	159	9.04	0
3	3	1.021	4.91	442	20.8	398	6.63	1
4	4	1.021	5.53	874	17.8	385	2.21	1

In [4]:
```
df.tail()
```

Out[4]:

	id	gravity	ph	osmo	cond	urea	calc	target
409	409	1.011	5.21	527	21.4	75	1.53	0
410	410	1.024	5.53	577	19.7	224	0.77	0
411	411	1.018	6.28	455	22.2	270	7.68	1
412	412	1.008	7.12	325	12.6	75	1.03	1
413	413	1.011	6.13	364	9.9	159	0.27	0

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

Out[5]:
```
(414, 8)
```

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

Out[6]:

	id	gravity	ph	osmo	cond	urea	calc	target
count	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000
mean	206.500000	1.017894	5.955459	651.545894	21.437923	278.657005	4.114638	0.444444
std	119.655756	0.006675	0.642260	234.676567	7.514750	136.442249	3.217641	0.497505
min	0.000000	1.005000	4.760000	187.000000	5.100000	10.000000	0.170000	0.000000
25%	103.250000	1.012000	5.530000	455.250000	15.500000	170.000000	1.450000	0.000000
50%	206.500000	1.018000	5.740000	679.500000	22.200000	277.000000	3.130000	0.000000
75%	309.750000	1.022000	6.280000	840.000000	27.600000	385.000000	6.630000	1.000000
max	413.000000	1.040000	7.940000	1236.000000	38.000000	620.000000	14.340000	1.000000

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 414 entries, 0 to 413
Data columns (total 8 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   id       414 non-null    int64
 1   gravity  414 non-null    float64
 2   ph       414 non-null    float64
 3   osmo     414 non-null    int64
 4   cond     414 non-null    float64
 5   urea     414 non-null    int64
 6   calc     414 non-null    float64
 7   target   414 non-null    int64
dtypes: float64(4), int64(4)
memory usage: 26.0 KB
```

In [8]:
```
df = df.drop_duplicates()
```

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
id         0
gravity    0
ph         0
osmo       0
cond       0
urea       0
calc       0
target     0
dtype: int64
```

In [10]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [11]:
```
df.nunique()
```

Out[11]:
```
id         414
gravity     28
ph          71
osmo        80
cond        65
urea        73
calc        85
target       2
dtype: int64
```

In [12]:
```
df['gravity'].value_counts
```

Out[12]:
```
<bound method IndexOpsMixin.value_counts of 0      1.013
1      1.025
2      1.009
3      1.021
4      1.021
       ...
409    1.011
410    1.024
411    1.018
412    1.008
413    1.011
Name: gravity, Length: 414, dtype: float64>
```

In [13]:
```
df['gravity'].unique()
```

Out[13]:
```
array([1.013, 1.025, 1.009, 1.021, 1.008, 1.015, 1.017, 1.007, 1.011,
       1.01 , 1.012, 1.018, 1.026, 1.02 , 1.024, 1.019, 1.034, 1.022,
       1.028, 1.014, 1.023, 1.029, 1.033, 1.005, 1.031, 1.04 , 1.006,
       1.027])
```

In [14]:
```
plt.figure(figsize=(15,6))
sns.countplot(x ='gravity',data=df,palette = 'hls')
plt.show()
```

In [15]:
```
df['osmo'].value_counts
```

Out[15]:
```
<bound method IndexOpsMixin.value_counts of 0      443
1      703
2      371
3      442
4      874
      ...
409    527
410    577
411    455
412    325
413    364
Name: osmo, Length: 414, dtype: int64>
```

In [16]:
```
df['osmo'].unique()
```

Out[16]:
```
array([ 443,  703,  371,  442,  874,  947,  450,  736,  527,  680,  364,
        541,  516,  251,  408,  760,  416, 1032,  395,  956,  704,  725,
        341,  779,  781,  668,  694,  242,  225,  840,  907,  461,  658,
        702, 1236,  456,  866,  631,  774,  822,  577,  312,  970, 1107,
        565,  343,  854,  386,  567,  698,  933,  543,  487,  721,  594,
        345,  876,  187,  283,  775,  253,  945,  559,  531,  325,  815,
        873,  684,  853,  241,  410,  838,  579,  679,  455,  256,  833,
        321,  803,  749], dtype=int64)
```

In [17]:
```
plt.figure(figsize=(15,6))
sns.countplot(x ='osmo',data=df,palette = 'hls')
plt.show()
```

In [18]:
```
df['ph'].value_counts
```

Out[18]:
```
<bound method IndexOpsMixin.value_counts of 0      6.19
1      5.40
2      6.13
3      4.91
4      5.53
       ...
409    5.21
410    5.53
411    6.28
412    7.12
413    6.13
Name: ph, Length: 414, dtype: float64>
```

In [19]:
```
df['ph'].unique()
```

Out[19]:
```
array([6.19, 5.4 , 6.13, 4.91, 5.53, 6.9 , 5.09, 7.38, 6.61, 6.81, 6.37,
       6.79, 6.56, 7.01, 6.63, 5.66, 5.35, 5.94, 5.87, 6.29, 6.75, 5.44,
       5.58, 7.61, 5.51, 5.41, 5.62, 5.98, 5.16, 5.68, 5.61, 5.6 , 6.03,
       7.92, 5.96, 6.14, 4.76, 6.28, 5.24, 6.52, 5.97, 5.47, 6.24, 7.2 ,
       5.85, 6.76, 5.86, 5.77, 5.56, 5.67, 6.27, 5.71, 6.88, 7.12, 5.74,
       7.3 , 6.3 , 5.64, 5.36, 5.33, 4.9 , 4.81, 7.9 , 5.9 , 5.21, 5.27,
       7.94, 5.73, 6.21, 6.58, 5.63])
```

In [20]:
```
plt.figure(figsize=(15,6))
sns.countplot(x ='ph',data=df,palette = 'hls')
plt.show()
```

In [21]:
```
plt.figure(figsize=(15,6))
sns.countplot(x ='cond',data=df,palette = 'hls')
plt.show()
```

In [22]:
```
plt.figure(figsize=(15,6))
sns.countplot(x ='urea',data=df,palette = 'hls')
plt.show()
```

In [23]:
```
plt.figure(figsize=(15,6))
sns.countplot(x ='calc',data=df,palette = 'hls')
plt.show()
```

In [24]:
```
plt.figure(figsize=(15,6))
sns.countplot(x ='target',data=df,palette = 'hls')
plt.show()
```

In [25]:
```
df = df.drop('id',axis = 1)
```

In [26]:
```
plt.figure(figsize=(30,20))
plt.pie(df['target'].value_counts(), labels=df['target'].value_counts().index, autopct='%1.1f%%', textprops={ '
                                        'color': 'black',
                                        'weight': 'bold',
                                        'family': 'serif' })
hfont = {'fontname':'serif', 'weight': 'bold'}
```

```
plt.title('Target', size=20, **hfont)
plt.show()
```

In [27]:

```
import plotly.express as px

fig = px.bar(df, x="target", y= df.index)
fig.show()
```

In [28]:
```
value_counts = df['target'].value_counts()
fig = px.pie(names=value_counts.index, values=value_counts.values)
fig.update_layout(
    title='Pie Chart of Target',
    title_x=0.5
)
fig.show()
```

In [29]:
```
for i in df.columns:
    if i != 'target':
        plt.figure(figsize=(15,6))
        sns.histplot(df[i],kde = True,bins =20,palette = 'hls')
        plt.xticks(rotation = 90)
        plt.show()
```

Count
gravity
0
10
20
30
40
50
60
1.005
1.010
1.015
1.020
1.025
1.030
1.035
1.040

Count
ph
0
10
20
30
40
50
60
70
80
5.0
5.5
6.0
6.5
7.0
7.5
8.0

Count
osmo
0
5
10
15
20
25
30
35
40
200
400
600
800
1000
1200

In [30]:
```
for i in df.columns:
    if i != 'target':
        plt.figure(figsize=(15,6))
        sns.distplot(df[i], kde = True, bins = 20)
        plt.xticks(rotation = 90)
        plt.show()
```

Density
gravity
0
10
20
30
40
50
60
70
80
1.00
1.01
1.02
1.03
1.04

Density
ph
0.0
0.2
0.4
0.6
0.8
1.0
1.2
4
5
6
7
8

Density
osmo
0.00000
0.00025
0.00050
0.00075
0.00100
0.00125
0.00150
0.00175
0.00200
0
200
400
600
800
1000
1200
1400

In [31]:
```
for i in df.columns:
    if i != 'target':
        plt.figure(figsize=(15,6))
        sns.boxplot(x=i, data = df, palette = 'hls')
        plt.xticks(rotation = 90)
        plt.show()
```

1.005
1.010
1.015
1.020
1.025
1.030
1.035
1.040
gravity

5.0
5.5
6.0
6.5
7.0
7.5
8.0
ph

200
400
600
800
1000
1200
osmo

In [83]:
```
for i in df.columns:
    if i != 'target':
        plt.figure(figsize=(15,6))
        sns.violinplot(x=i, data = df, palette = 'hls')
        plt.xticks(rotation = 90)
        plt.show()
```

1.00
1.01
1.02
1.03
1.04
gravity

4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
ph

0
200
400
600
800
1000
1200
1400
osmo

In [84]:
```
for i in df.columns:
    if i != 'target':
        fig = px.histogram(df[i], nbins=20, color_discrete_sequence=['#636EFA'],
                           marginal='box', opacity=0.7)
        fig.update_layout(
            title=f'Histogram of {i}',
            xaxis_title=i,
            yaxis_title='Count',
            showlegend=False
        )
        fig.show()
```

Histogram of gravity
Count
80
60
40
20
0
1.01
1.02
1.03
1.04
gravity

In [85]:

```
for i in df.columns:
    if i != 'target':
        fig = px.histogram(df, x=i, color_discrete_sequence=['#636EFA'], nbins=20, opacity=0.7,
                           title=f'Histogram of {i}', labels={'x': i, 'count': 'Frequency'},
                           template='plotly_white', width=800, height=400,
                           marginal='rug', barmode='overlay',
                           hover_data=[df['target']])
        fig.update_layout(xaxis=dict(type='category', title=i,
                                     tickangle=-90, tickfont=dict(size=10)),
                          yaxis_title='Frequency', showlegend=False)
        fig.show()
```

In [86]:

```
for i in df.columns:
    if i != 'target':
        fig = px.box(df, x=df[i], color_discrete_sequence=['#636EFA'])
        fig.update_layout(
            title='Box Plot of ' + i,
            xaxis_title=i,
            yaxis_title='Value',
        )
        fig.show()
```

In [87]:
```
for i in df.columns:
    if i != 'target':
        fig = px.violin(df, y=i, x='target', box=True, points="all", hover_data=df.columns)
        fig.update_layout(title=i, xaxis_title="Target", yaxis_title=i)
        fig.show()
```

In [88]:

```
for i in df.columns:
    if i != 'target':
        plt.figure(figsize=(15,6))
        sns.barplot(x = df['target'], y = df[i], data = df, ci = None, palette = 'hls')
        plt.xticks(rotation = 90)
        plt.show()
```

gravity
1.0
0.8
0.6
0.4
0.2
0.0
0
1
target

ph
6
5
4
3
2
1
0
0
1
target

osmo
700
600
500
400
300
200
100
0
0
1
target

In [89]:
```
for i in df.columns:
    if i != 'target':
        fig = px.bar(df, x="target", y=i, color="target", barmode="group")
        fig.update_layout(
            title=f"{i} by Target",
            xaxis_title="Target",
            yaxis_title=i,
            legend_title="Target"
        )
        fig.show()
```

In [90]:

```
for i in df.columns:
    for j in df.columns:
        if i != j:
            if i != 'target':
                plt.figure(figsize=(15,6))
                sns.lineplot(x = df[i], y = df[j], data = df, ci = None, palette = 'hls')
                plt.xticks(rotation = 90)
                plt.show()
```

ph
6.8
6.6
6.4
6.2
6.0
5.8
5.6
5.4
5.2
1.005
1.010
1.015
1.020
1.025
1.030
1.035
1.040
gravity

osmo
1000
900
800
700
600
500
400
300
200
1.005
1.010
1.015
1.020
1.025
1.030
1.035
1.040
gravity

cond
27.5
25.0
22.5
20.0
17.5
15.0
12.5
10.0
1.005
1.010
1.015
1.020
1.025
1.030
1.035
1.040
gravity

urea
gravity

calc
gravity

target
gravity

1.030
1.025
1.020
1.015
1.010
gravity
5.0
5.5
6.0
6.5
7.0
7.5
8.0
ph

1200
1000
800
600
400
osmo
5.0
5.5
6.0
6.5
7.0
7.5
8.0
ph

27.5
25.0
22.5
20.0
17.5
15.0
12.5
10.0
cond
5.0
5.5
6.0
6.5
7.0
7.5
8.0
ph

urea
ph

calc
ph

target
ph

1.030
1.025
1.020
1.015
1.010
1.005
gravity
200
400
600
800
1000
1200
osmo

7.0
6.5
6.0
5.5
5.0
ph
200
400
600
800
1000
1200
osmo

35
30
25
20
15
10
5
cond
200
400
600
800
1000
1200
osmo

urea
osmo

calc
osmo

target
osmo

gravity
1.030
1.025
1.020
1.015
1.010
5
10
15
20
25
30
35
cond

ph
7.00
6.75
6.50
6.25
6.00
5.75
5.50
5.25
5
10
15
20
25
30
35
cond

osmo
1000
800
600
400
200
5
10
15
20
25
30
35
cond

urea
cond

calc
cond

target
cond

gravity
1.035
1.030
1.025
1.020
1.015
1.010
1.005
0
100
200
300
400
500
600
urea

ph
6.5
6.0
5.5
5.0
0
100
200
300
400
500
600
urea

osmo
1200
1000
800
600
400
200
0
100
200
300
400
500
600
urea

cond
urea

calc
urea

target
urea

gravity
1.040
1.035
1.030
1.025
1.020
1.015
1.010
1.005
0
2
4
6
8
10
12
14
calc

ph
6.75
6.50
6.25
6.00
5.75
5.50
5.25
0
2
4
6
8
10
12
14
calc

osmo
1000
900
800
700
600
500
400
300
0
2
4
6
8
10
12
14
calc

In [91]:
```
for i in df.columns:
    for j in df.columns:
        if i != j:
            if i != 'target':
                plt.figure(figsize=(15,6))
                sns.scatterplot(x = df[i], y = df[j], data = df, style = None, palette = 'hls')
                plt.xticks(rotation = 90)
                plt.show()
```

ph
8.0
7.5
7.0
6.5
6.0
5.5
5.0
1.005
1.010
1.015
1.020
1.025
1.030
1.035
1.040
gravity

osmo
1200
1000
800
600
400
200
1.005
1.010
1.015
1.020
1.025
1.030
1.035
1.040
gravity

cond
35
30
25
20
15
10
5
1.005
1.010
1.015
1.020
1.025
1.030
1.035
1.040
gravity

urea
gravity

calc
gravity

target
gravity

gravity
1.040
1.035
1.030
1.025
1.020
1.015
1.010
1.005
5.0
5.5
6.0
6.5
7.0
7.5
8.0
ph

osmo
1200
1000
800
600
400
200
5.0
5.5
6.0
6.5
7.0
7.5
8.0
ph

cond
35
30
25
20
15
10
5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
ph

urea
ph

calc
ph

target
ph

gravity
1.040
1.035
1.030
1.025
1.020
1.015
1.010
1.005
200
400
600
800
1000
1200
osmo

ph
8.0
7.5
7.0
6.5
6.0
5.5
5.0
200
400
600
800
1000
1200
osmo

cond
35
30
25
20
15
10
5
200
400
600
800
1000
1200
osmo

urea
osmo

calc
osmo

target
osmo

gravity
1.040
1.035
1.030
1.025
1.020
1.015
1.010
1.005
5
10
15
20
25
30
35
cond

ph
8.0
7.5
7.0
6.5
6.0
5.5
5.0
5
10
15
20
25
30
35
cond

osmo
1200
1000
800
600
400
200
5
10
15
20
25
30
35
cond

urea
cond

calc
cond

target
cond

gravity
urea

ph
urea

osmo
urea

cond
urea

calc
urea

target
urea

gravity
1.040
1.035
1.030
1.025
1.020
1.015
1.010
1.005
0
2
4
6
8
10
12
14
calc

ph
8.0
7.5
7.0
6.5
6.0
5.5
5.0
0
2
4
6
8
10
12
14
calc

osmo
1200
1000
800
600
400
200
0
2
4
6
8
10
12
14
calc

In [92]:
```
for i in df.columns:
    for j in df.columns:
        if i != j:
            if i != 'target':
                fig = px.line(df, x=i, y=j, color='target', template='plotly_dark')
                fig.update_layout(
                    title=f'{j} vs {i}',
                    xaxis_title=i,
                    yaxis_title=j
                )
                fig.show()
```

In [93]:

```
for i in df.columns:
    for j in df.columns:
        if i != j:
            if i != 'target':
                fig = px.scatter(df, x=i, y=j, color='target', color_discrete_sequence=['#636EFA', '#EF553B'])
                fig.update_layout(title=f'Scatter Plot of {i} vs {j}', xaxis_title=i, yaxis_title=j)
                fig.show()
```

In [94]:
```
import plotly.graph_objs as go
```

In [95]:
```
fig = go.Figure(data=[go.Scatter3d(
    x=df['gravity'],
    y=df['ph'],
    z=df['osmo'],
    mode='markers',
    marker=dict(
        size=5,
        color=df['target'],
        colorscale='Viridis',
        opacity=0.8
    )
)])

fig.update_layout(
    title='3D Scatter Plot',
    scene=dict(
        xaxis_title='Gravity',
        yaxis_title='pH',
        zaxis_title='Osmo'
```

```
    )
)

fig.show()
```

In [96]:

```
plt.figure(figsize=(15,6))
sns.pairplot(data = df, palette = 'hls')
plt.xticks(rotation = 90)
plt.show()
```

```
<Figure size 1500x600 with 0 Axes>
```

In [97]:
```
from pandas.plotting import scatter_matrix
scatter_matrix(df, diagonal='hist', color='b', alpha=0.3, figsize=(15,6))
fig = go.Figure(data=px.scatter_matrix(df, color='target', dimensions=['gravity', 'ph', 'osmo', 'cond', 'urea',
fig.update_layout(title='Pairplot of Features', width=800, height=800)
fig.show()
```

In [98]:
```
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
gravity      0.01
ph           0.75
osmo       384.75
cond        12.10
urea       215.00
calc         5.18
target       1.00
dtype: float64
```

In [99]:
```
df_new = df[~((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))).any(axis = 1)]
```

In [100…
```
df_new.shape
```

Out[100]:
```
(399, 7)
```

In [101…
```
from scipy.stats import ttest_ind
```

In [102…
```
group1 = df[df['target'] == 0]['urea']
group2 = df[df['target'] == 1]['urea']
```

In [ ]:

In [ ]: