In [1]:

```
#Load Data into DataFrame
import pandas as pd
df = pd.read_csv('diagnosis.csv')
df
```

Out[1]:

	TempP	Nausea	Lumbar	Pushing	Micturition	Burning	inflammation	Nephritis
0	35.5	no	yes	no	no	no	no	no
1	35.9	no	no	yes	yes	yes	yes	no
2	35.9	no	yes	no	no	no	no	no
3	36.0	no	no	yes	yes	yes	yes	no
4	36.0	no	yes	no	no	no	no	no
...	...	...	...	...	...	...	...	...
115	41.4	no	yes	yes	no	yes	no	yes
116	41.5	no	no	no	no	no	no	no
117	41.5	yes	yes	no	yes	no	no	yes
118	41.5	no	yes	yes	no	yes	no	yes
119	41.5	no	yes	yes	no	yes	no	yes

120 rows × 8 columns

In [2]:

```
#Convert all yes and no's to 1 and 0
df['Nausea'] = df.Nausea.map(dict(yes=1, no=0))
df['Lumbar'] = df.Lumbar.map(dict(yes=1, no=0))
df['Pushing'] = df.Pushing.map(dict(yes=1, no=0))
df['Micturition'] = df.Micturition.map(dict(yes=1, no=0))
df['Burning'] = df.Burning.map(dict(yes=1, no=0))
df['inflammation'] = df.inflammation.map(dict(yes=1, no=0))
df['Nephritis'] = df.Nephritis.map(dict(yes=1, no=0))
df
```

Out[2]:

	TempP	Nausea	Lumbar	Pushing	Micturition	Burning	inflammation	Nephritis
0	35.5	0	1	0	0	0	0	0
1	35.9	0	0	1	1	1	1	0
2	35.9	0	1	0	0	0	0	0
3	36.0	0	0	1	1	1	1	0
4	36.0	0	1	0	0	0	0	0
...	...	...	...	...	...	...	...	...
115	41.4	0	1	1	0	1	0	1
116	41.5	0	0	0	0	0	0	0
117	41.5	1	1	0	1	0	0	1
118	41.5	0	1	1	0	1	0	1
119	41.5	0	1	1	0	1	0	1

120 rows × 8 columns

In [3]:

```
# Load libraries
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
...
#Split data into features and target variables
X = df[['TempP', 'Nausea', 'Lumbar', 'Pushing', 'Micturition', 'Burning']]
y1 = df['inflammation']
y2 = df['Nephritis']
#Define test and train datasets

X_train, X_validation, Y_train, Y_validation = train_test_split(X, y1, test_size=0.20, random_state=1)
...
print(y1)
#Fit data to inflammation model
inf_models = SVC()
inf_models.fit(X_train,Y_train)
```

```
0      0
1      1
2      0
3      1
4      0
      ..
115    0
116    0
117    0
118    0
119    0
Name: inflammation, Length: 120, dtype: int64
```

Out[3]:
```
SVC()
```

In [4]:

```
# Load libraries
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
...
#Define test and train datasets
X_train, X_validation1, Y_train1, Y_validation1 = train_test_split(X, y2, test_size=0.20, random_state=1)
...
# Spot Check Algorithms
models2 = SVC(gamma='auto')
models2.fit(X_train,Y_train)
#models2.append(('SVM', SVC(gamma='auto')))
# evaluate each model in turn

from sklearn.model_selection import GridSearchCV

# define parameter range
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf']}
#Use gridsearchCV to improve the hyperplane of the svc model
grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)

# fitting the model for grid search
grid.fit(X_train, Y_train1)
```

```
[CV 2/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.842 total time=   0.0s
[CV 3/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.789 total time=   0.0s
[CV 4/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.895 total time=   0.0s
[CV 5/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.895 total time=   0.0s
[CV 1/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.650 total time=   0.0s
[CV 2/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.684 total time=   0.0s
[CV 3/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.632 total time=   0.0s
[CV 4/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.632 total time=   0.0s
[CV 5/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.632 total time=   0.0s
[CV 1/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.650 total time=   0.0s
[CV 2/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.684 total time=   0.0s
[CV 3/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.632 total time=   0.0s
[CV 4/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.632 total time=   0.0s
[CV 5/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.632 total time=   0.0s
[CV 1/5] END .........C=10, gamma=1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 2/5] END .........C=10, gamma=1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 3/5] END .........C=10, gamma=1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 4/5] END .........C=10, gamma=1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 5/5] END .........C=10, gamma=1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 1/5] END .......C=10, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
```

In [5]:

```
#Predicted Inflammation values
y_pred = inf_models.predict(X_validation)
#Predicted Nephritis values
y_pred1 = models2.predict(X_validation1)
#Predicted Inflammation values AFTER Grid Search
grid_predictions = grid.predict(X_validation)
from sklearn.metrics import confusion_matrix
#Comparing Confusion Matricies
print('Inflammation confusion matrix: ')

print(confusion_matrix(Y_validation, y_pred))
print('Nephritis confusion matrix: ')

print(confusion_matrix(Y_validation1, y_pred1))
print('Inflammation confusion matrix AFTER GridSearchCV: ')

print(confusion_matrix(Y_validation, grid_predictions))
```

```
Inflammation confusion matrix:
[[12  0]
 [12  0]]
Nephritis confusion matrix:
[[ 1  7]
 [11  5]]
Inflammation confusion matrix AFTER GridSearchCV:
[[ 1 11]
 [ 7  5]]
```

In [6]:

```
#Comparing Classification Reports
print('Inflammation Classification Report: ')
print(classification_report(Y_validation, y_pred))
print('Inflammation Classification Report AFTER GridSearchCV: ')
print(classification_report(Y_validation, grid_predictions))
print('Nephritis Classification Report: ')
print(classification_report(Y_validation1, y_pred1))
```

```
Inflammation Classification Report:
              precision    recall  f1-score   support

           0       0.50      1.00      0.67        12
           1       0.00      0.00      0.00        12

    accuracy                           0.50        24
   macro avg       0.25      0.50      0.33        24
weighted avg       0.25      0.50      0.33        24

Inflammation Classification Report AFTER GridSearchCV:
              precision    recall  f1-score   support

           0       0.12      0.08      0.10        12
           1       0.31      0.42      0.36        12

    accuracy                           0.25        24
   macro avg       0.22      0.25      0.23        24
weighted avg       0.22      0.25      0.23        24
```

In [7]:

```
X_train, X_validation, Y_train, Y_validation = train_test_split(X, y1, test_size=0.20, random_state=1)
...
# Spot Check Algorithms
models = []
models.append(('SVM', SVC(gamma='auto')))
# evaluate each model in turn

kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
cv_results = cross_val_score(grid, X_train, Y_train, cv=kfold, scoring='accuracy')
results.append(cv_results)
names.append(name)
print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))
```

```
Fitting 5 folds for each of 25 candidates, totalling 125 fits
[CV 1/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.944 total time=   0.0s
[CV 2/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.765 total time=   0.0s
[CV 3/5] END ........C=0.1, gamma=1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 4/5] END ........C=0.1, gamma=1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 5/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.824 total time=   0.0s
[CV 1/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.722 total time=   0.0s
[CV 2/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.647 total time=   0.0s
[CV 3/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.765 total time=   0.0s
[CV 4/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.824 total time=   0.0s
[CV 5/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.824 total time=   0.0s
[CV 1/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.500 total time=   0.0s
[CV 2/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.529 total time=   0.0s
[CV 3/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.529 total time=   0.0s
[CV 4/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.529 total time=   0.0s
[CV 5/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.471 total time=   0.0s
[CV 1/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.500 total time=   0.0s
[CV 2/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.529 total time=   0.0s
[CV 3/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.529 total time=   0.0s
```

In [8]:
```
...
# class distribution
print(df.groupby('inflammation').size())
...
# class distribution
print(df.groupby('Nephritis').size())
```

```
inflammation
0    61
1    59
dtype: int64
Nephritis
0    70
1    50
dtype: int64
```

In [9]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [12]:
```
#descriptive statistics
df.describe()
```

Out[12]:

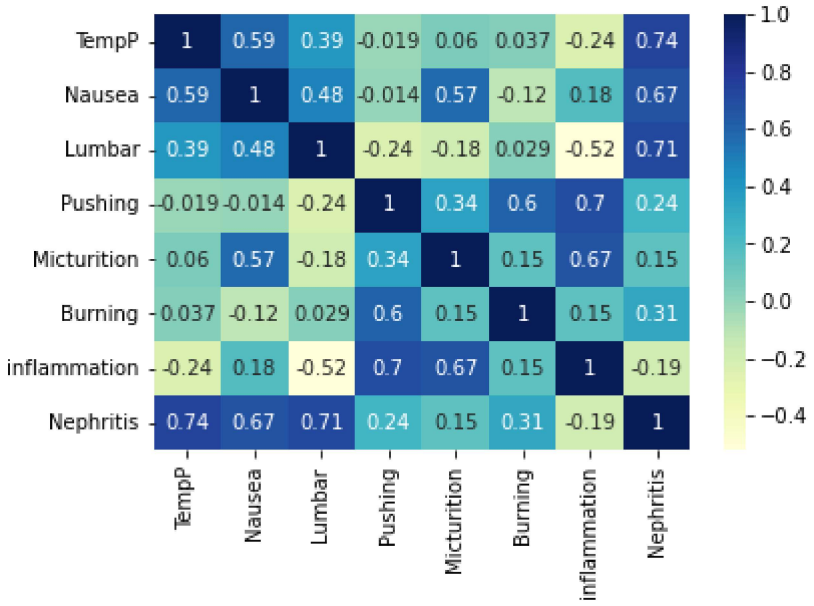
	TempP	Nausea	Lumbar	Pushing	Micturition	Burning	inflammation	Nephritis
count	120.000000	120.000000	120.000000	120.000000	120.000000	120.000000	120.000000	120.000000
mean	38.724167	0.241667	0.583333	0.666667	0.491667	0.416667	0.491667	0.416667
std	1.819132	0.429888	0.495074	0.473381	0.502027	0.495074	0.502027	0.495074
min	35.500000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	37.100000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	37.950000	0.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000
75%	40.600000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
max	41.500000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

In [10]:
```
df.corr()
```

Out[10]:

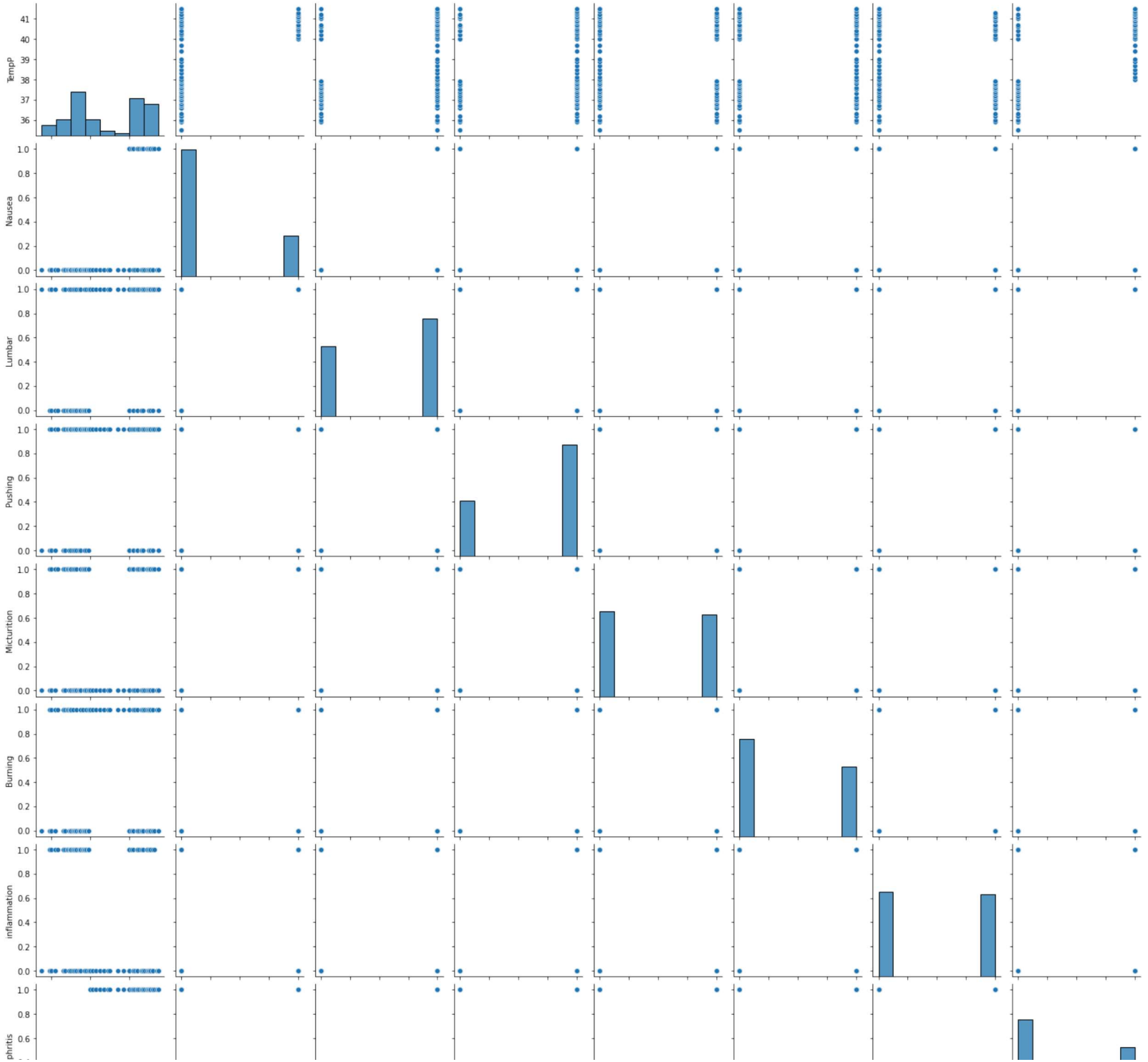
	TempP	Nausea	Lumbar	Pushing	Micturition	Burning	inflammation	Nephritis
TempP	1.000000	0.593152	0.391971	-0.018866	0.060493	0.037245	-0.236718	0.737055
Nausea	0.593152	1.000000	0.477105	-0.013765	0.574007	-0.121744	0.184630	0.667947
Lumbar	0.391971	0.477105	1.000000	-0.239046	-0.183142	0.028571	-0.521251	0.714286
Pushing	-0.018866	-0.013765	-0.239046	1.000000	0.341816	0.597614	0.695418	0.239046
Micturition	0.060493	0.574007	-0.183142	0.341816	1.000000	0.149331	0.666574	0.149331
Burning	0.037245	-0.121744	0.028571	0.597614	0.149331	1.000000	0.149331	0.314286
inflammation	-0.236718	0.184630	-0.521251	0.695418	0.666574	0.149331	1.000000	-0.188777
Nephritis	0.737055	0.667947	0.714286	0.239046	0.149331	0.314286	-0.188777	1.000000

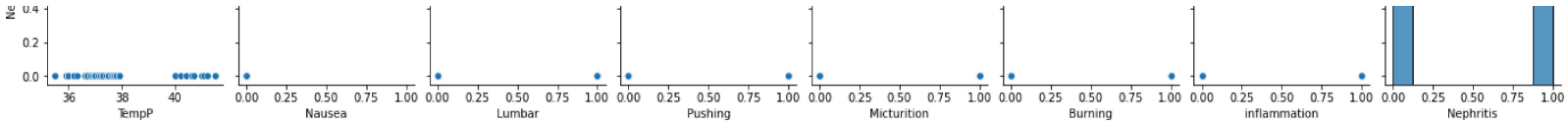
In [17]:
```
#visualizing correlation between features
dataplot = sns.heatmap(df.corr(), cmap="YlGnBu", annot=True)
```



In [19]:
```
#visualize scatterplot matrix
sns.pairplot(df)
```

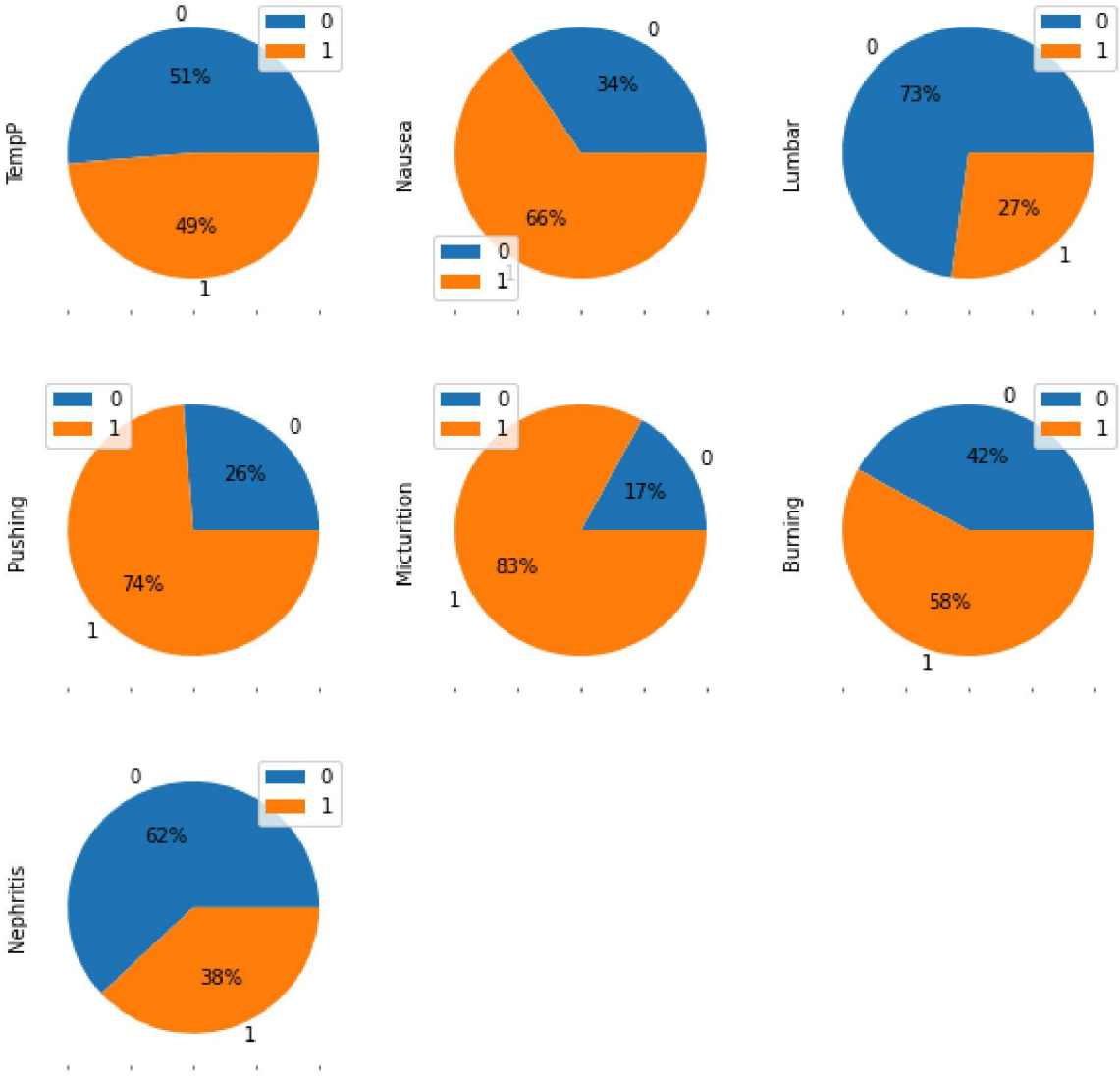
Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x2ba2401dbb0>
```


TempF
Nausea
Lumbar
Pushing
Micturition
Burning
inflammation
phritis


Ne
0.4
0.2
0.0
36
38
40
TempP
0.00
0.25
0.50
0.75
1.00
Nausea
Lumbar
Pushing
Micturition
Burning
inflammation
Nephritis

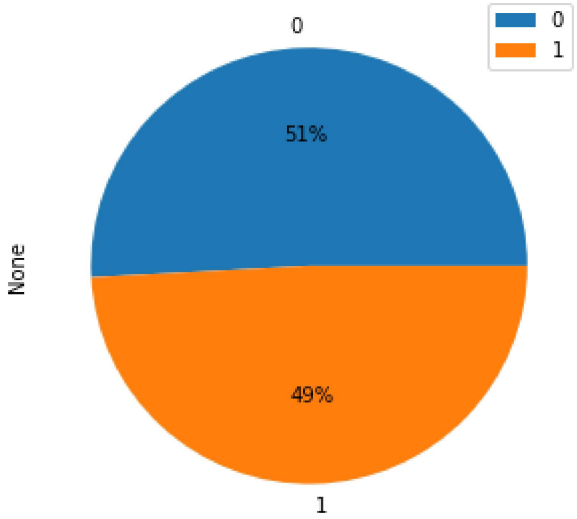
In [53]:
```
#view features by inflammation decision
df.groupby(['inflammation']).sum().plot(
    kind='pie', subplots=True, autopct='%1.0f%%', figsize=(10,10), layout=(3,3))
```

Out[53]:
```
array([[<AxesSubplot:ylabel='TempP'>, <AxesSubplot:ylabel='Nausea'>,
        <AxesSubplot:ylabel='Lumbar'>],
       [<AxesSubplot:ylabel='Pushing'>,
        <AxesSubplot:ylabel='Micturition'>,
        <AxesSubplot:ylabel='Burning'>],
       [<AxesSubplot:ylabel='Nephritis'>, <AxesSubplot:>, <AxesSubplot:>]],
      dtype=object)
```


TempP
0
1
51%
49%
Nausea
34%
66%
Lumbar
73%
27%
Pushing
26%
74%
Micturition
17%
83%
Burning
42%
58%
Nephritis
62%
38%

In [48]:
```
#inflammation decision
df.groupby('inflammation').size().plot(kind='pie', legend=True, autopct='%1.0f%%')
```

Out[48]: <AxesSubplot:ylabel='None'>



In [ ]: