In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, classification_report
import pickle
```

In [2]:
```
train = pd.read_csv("Disease_Prediction_Training.csv")
test = pd.read_csv("Disease_Prediction_Testing.csv")
```

In [3]:
```
train
```

Out[3]:

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity
0	1	1	1	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0
2	1	0	1	0	0	0	0	0	0
3	1	1	0	0	0	0	0	0	0
4	1	1	1	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
4915	0	0	0	0	0	0	0	0	0
4916	0	1	0	0	0	0	0	0	0
4917	0	0	0	0	0	0	0	0	0
4918	0	1	0	0	0	0	1	0	0
4919	0	1	0	0	0	0	0	0	0

4920 rows × 134 columns

In [4]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4920 entries, 0 to 4919
Columns: 134 entries, itching to Unnamed: 133
dtypes: float64(1), int64(132), object(1)
memory usage: 5.0+ MB
```

In [5]:
```
train.isnull().sum()
```

Out[5]:
```
itching                 0
skin_rash               0
nodal_skin_eruptions    0
continuous_sneezing     0
shivering               0
                       ...
blister                 0
red_sore_around_nose    0
yellow_crust_ooze       0
prognosis               0
Unnamed: 133         4920
Length: 134, dtype: int64
```

In [6]:
```
train = train.drop(columns=['Unnamed: 133'])
```

In [7]:
```
train.isnull().sum()
```

Out[7]:
```
itching                 0
skin_rash               0
nodal_skin_eruptions    0
continuous_sneezing     0
shivering               0
                       ..
inflammatory_nails      0
blister                 0
red_sore_around_nose    0
yellow_crust_ooze       0
prognosis               0
Length: 133, dtype: int64
```

In [8]:
```
train.isnull().sum().sum()
```

Out[8]:
```
0
```

In [9]:
```
train
```

Out[9]:

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity
0	1	1	1	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0
2	1	0	1	0	0	0	0	0	0
3	1	1	0	0	0	0	0	0	0
4	1	1	1	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
4915	0	0	0	0	0	0	0	0	0
4916	0	1	0	0	0	0	0	0	0
4917	0	0	0	0	0	0	0	0	0
4918	0	1	0	0	0	0	1	0	0
4919	0	1	0	0	0	0	0	0	0

4920 rows × 133 columns

In [10]:
```
train.describe()
```

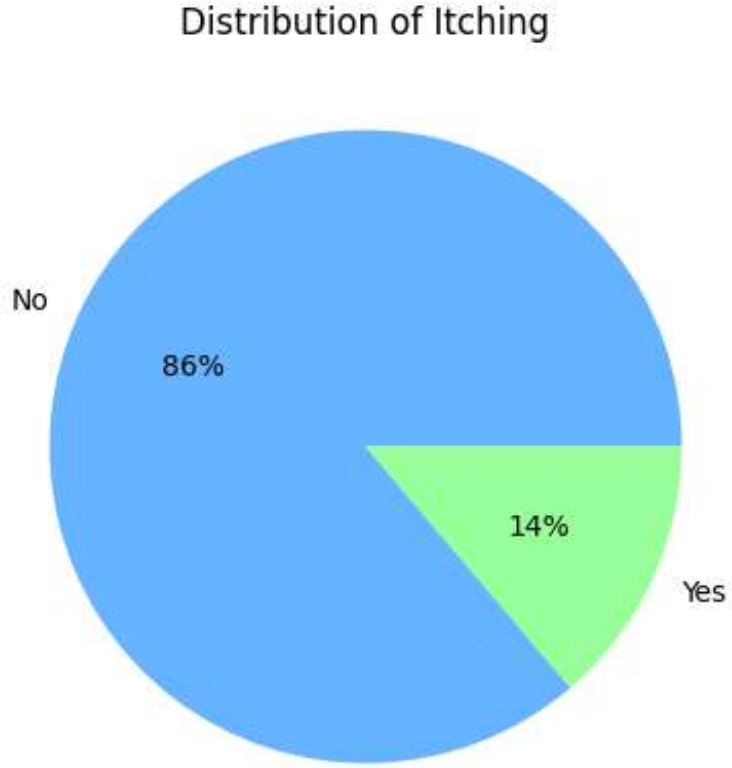
Out[10]:

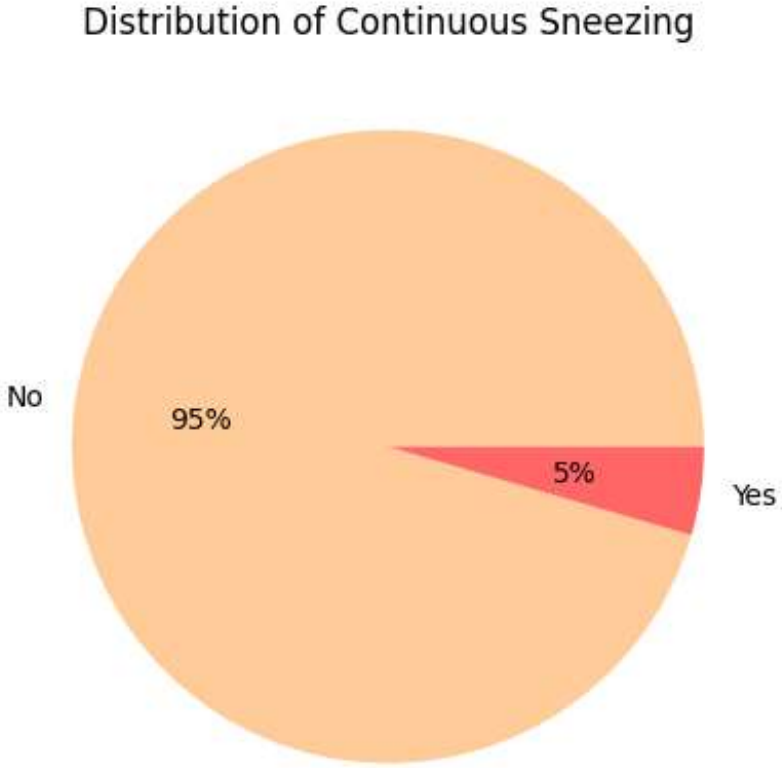
	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	sto
count	4920.000000	4920.000000	4920.000000	4920.000000	4920.000000	4920.000000	4920.000000	4
mean	0.137805	0.159756	0.021951	0.045122	0.021951	0.162195	0.139024	
std	0.344730	0.366417	0.146539	0.207593	0.146539	0.368667	0.346007	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

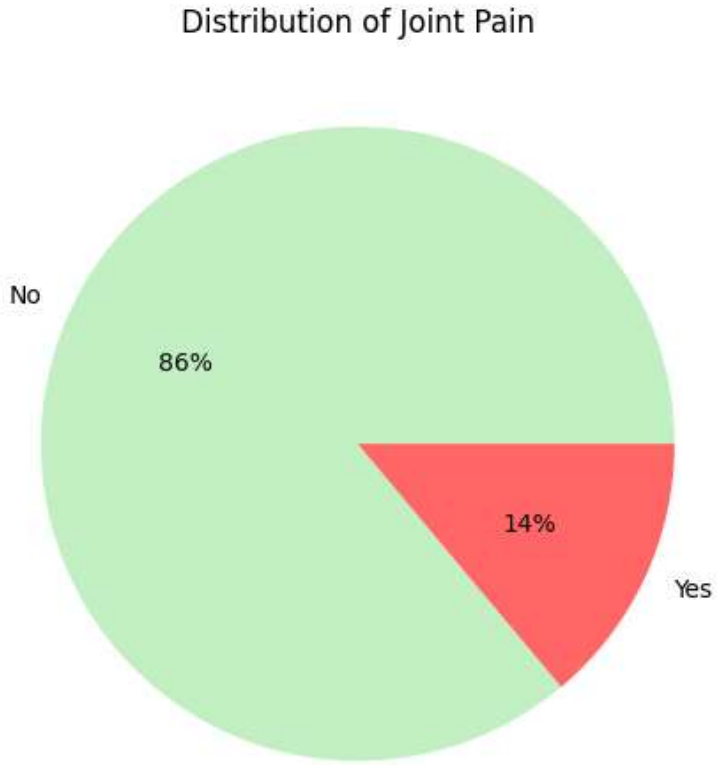
8 rows × 132 columns

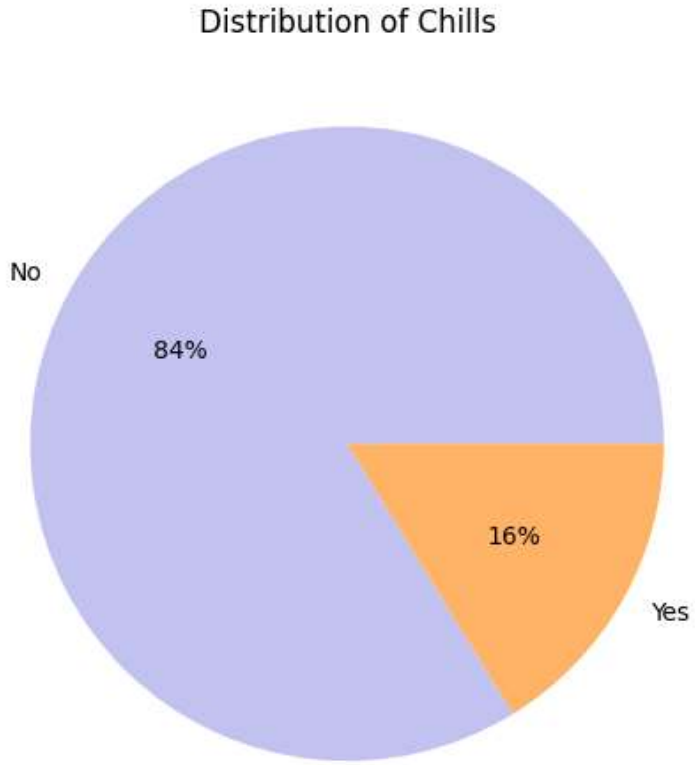
In [11]:

```
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
itching_counts = train['itching'].value_counts()
colors_itching = ['#66b3ff', '#99ff99']
plt.pie(x=itching_counts, labels=['No', 'Yes'], autopct='%.0f%%', colors=colors_itching)
plt.title("Distribution of Itching")

plt.subplot(1, 2, 2)
sneezing_counts = train['continuous_sneezing'].value_counts()
colors_sneezing = ['#ffcc99', '#ff6666']
plt.pie(x=sneezing_counts, labels=['No', 'Yes'], autopct='%.0f%%', colors=colors_sneezing)
plt.title("Distribution of Continuous Sneezing")

plt.figure(figsize=(18, 5))
plt.subplot(1, 3, 1)
joint_pain_counts = train['joint_pain'].value_counts()
colors_joint_pain = ['#c2f0c2', '#ff6666']
plt.pie(x=joint_pain_counts, labels=['No', 'Yes'], autopct='%.0f%%', colors=colors_joint_pain)
plt.title("Distribution of Joint Pain")

plt.subplot(1, 3, 2)
chills_counts = train['chills'].value_counts()
colors_chills = ['#c2c2f0', '#ffb366']
plt.pie(x=chills_counts, labels=['No', 'Yes'], autopct='%.0f%%', colors=colors_chills)
plt.title("Distribution of Chills")

plt.tight_layout()
plt.show()
```


Distribution of Itching
No
86%
14%
Yes


Distribution of Continuous Sneezing
No
95%
5%
Yes

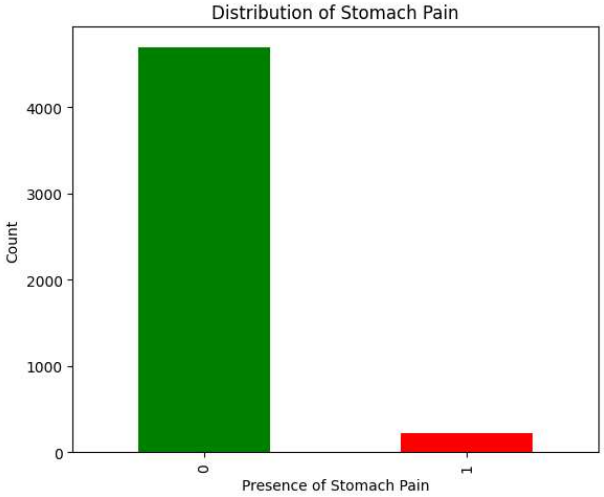

Distribution of Joint Pain
No
86%
14%
Yes

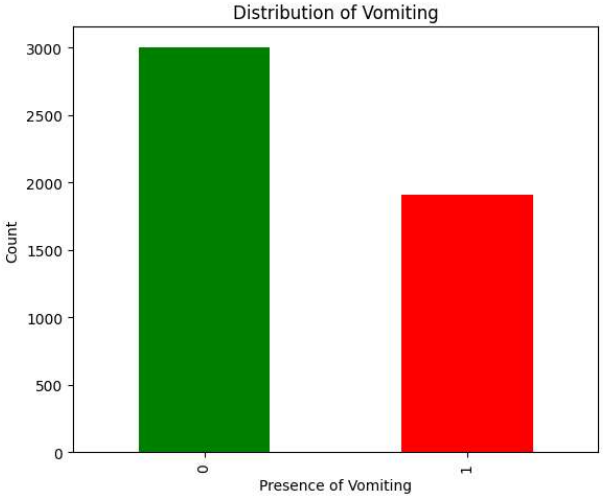

Distribution of Chills
No
84%
16%
Yes

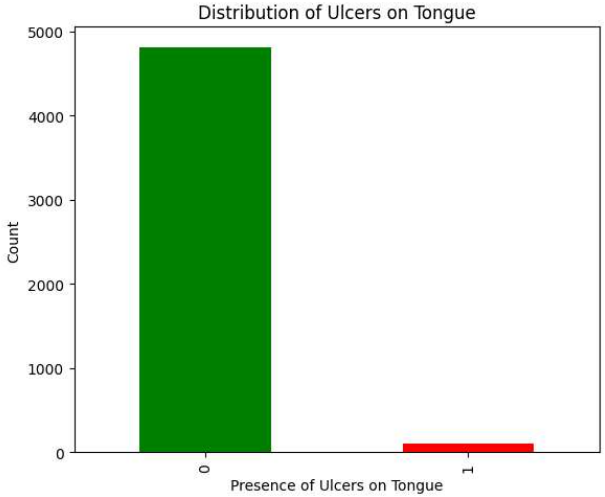
In [12]:

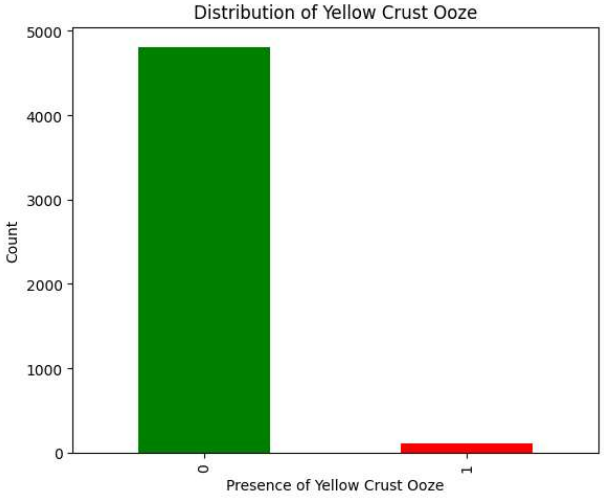
```
plt.figure(figsize=(18, 10))

# Bar chart for 'stomach_pain'
plt.subplot(2, 3, 1)
train['stomach_pain'].value_counts().plot(kind='bar', color=['g', 'r'])
plt.title("Distribution of Stomach Pain")
plt.xlabel("Presence of Stomach Pain")
plt.ylabel("Count")

# Bar chart for 'vomiting'
plt.subplot(2, 3, 2)
train['vomiting'].value_counts().plot(kind='bar', color=['g', 'r'])
plt.title("Distribution of Vomiting")
plt.xlabel("Presence of Vomiting")
plt.ylabel("Count")

# Bar chart for 'ulcers_on_tongue'
plt.subplot(2, 3, 3)
train['ulcers_on_tongue'].value_counts().plot(kind='bar', color=['g', 'r'])
plt.title("Distribution of Ulcers on Tongue")
plt.xlabel("Presence of Ulcers on Tongue")
plt.ylabel("Count")

# Bar chart for 'yellow_crust_ooze'
plt.subplot(2, 3, 4)
train['yellow_crust_ooze'].value_counts().plot(kind='bar', color=['g', 'r'])
plt.title("Distribution of Yellow Crust Ooze")
plt.xlabel("Presence of Yellow Crust Ooze")
plt.ylabel("Count")

# Bar chart for 'joint_pain'
plt.subplot(2, 3, 5)
train['joint_pain'].value_counts().plot(kind='bar', color=['g', 'r'])
plt.title("Distribution of Joint Pain")
```

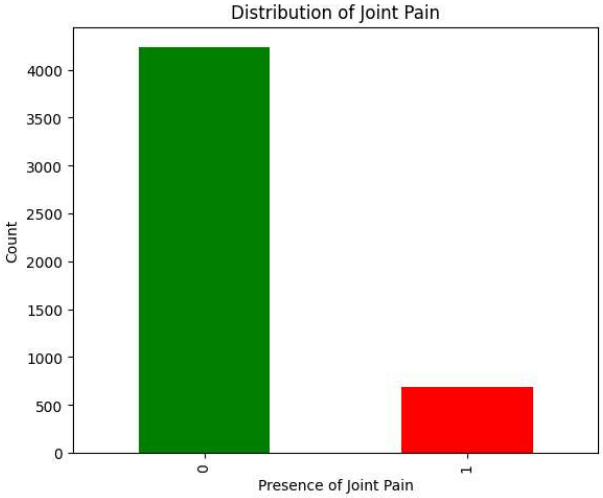
```
plt.xlabel("Presence of Joint Pain")
plt.ylabel("Count")

# Bar chart for 'inflammatory_nails'
plt.subplot(2, 3, 6)
train['inflammatory_nails'].value_counts().plot(kind='bar', color=['g', 'r'])
plt.title("Distribution of Inflammatory Nails")
plt.xlabel("Presence of Inflammatory Nails")
plt.ylabel("Count")

plt.tight_layout(h_pad=2.5, w_pad=2.5)
plt.show()
```

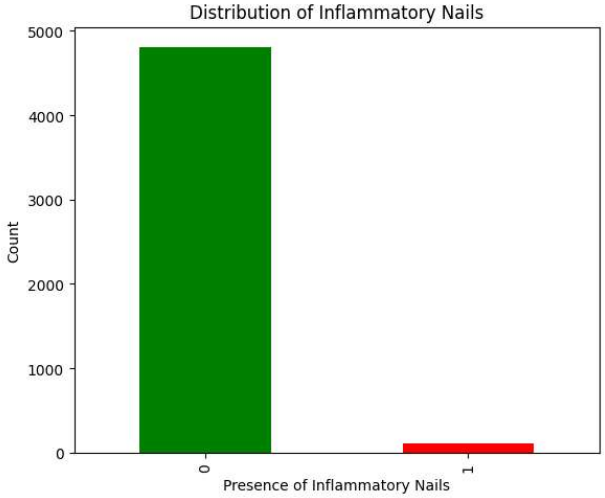

Distribution of Stomach Pain
Count
Presence of Stomach Pain
0
1
0
1000
2000
3000
4000


Distribution of Vomiting
Count
Presence of Vomiting
0
1
0
500
1000
1500
2000
2500
3000


Distribution of Ulcers on Tongue
Count
Presence of Ulcers on Tongue
0
1
0
1000
2000
3000
4000
5000

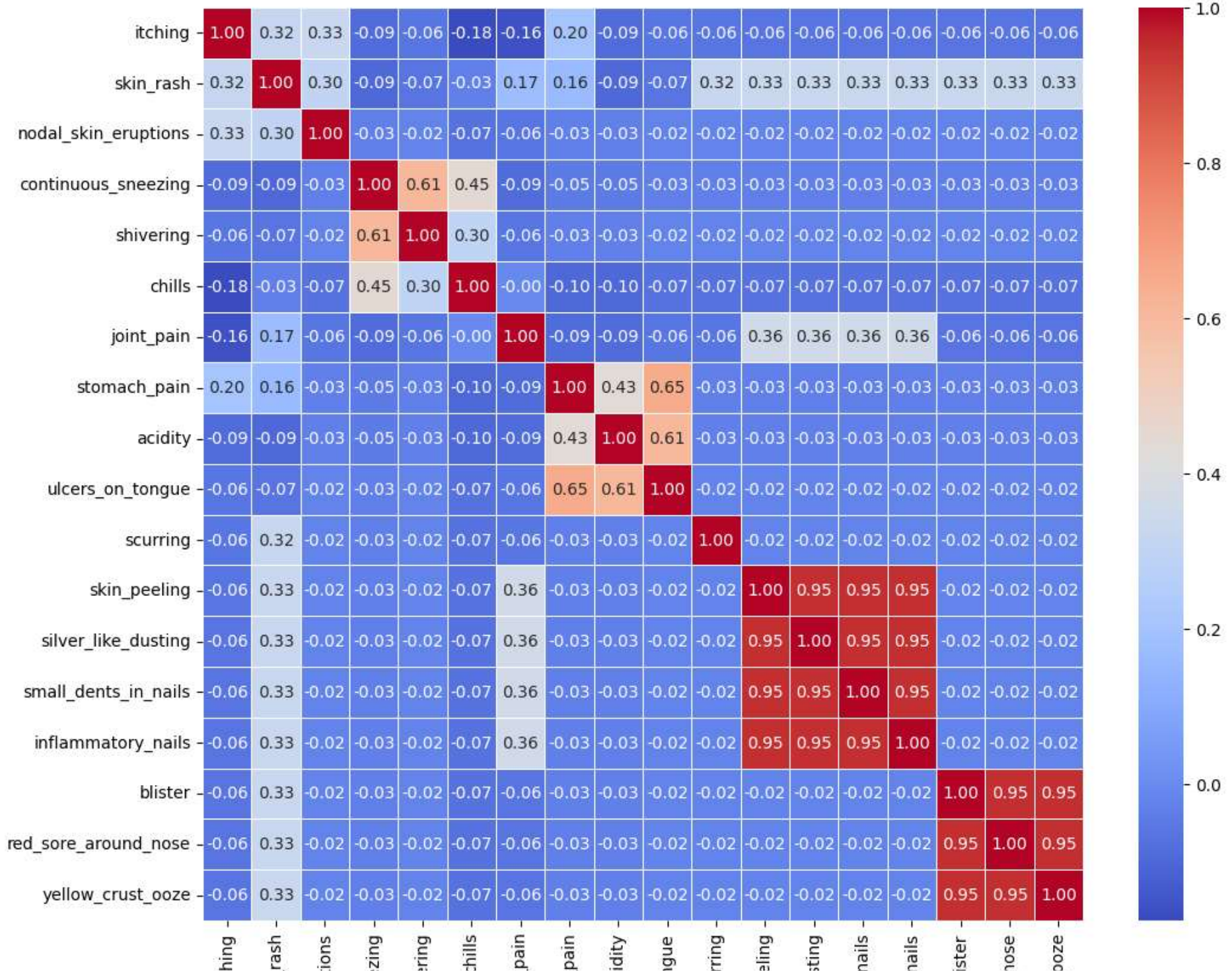

Distribution of Yellow Crust Ooze
Count
Presence of Yellow Crust Ooze
0
1
0
1000
2000
3000
4000
5000


Distribution of Joint Pain
Count
Presence of Joint Pain
0
1
0
500
1000
1500
2000
2500
3000
3500
4000


Distribution of Inflammatory Nails
Count
Presence of Inflammatory Nails
0
1
0
1000
2000
3000
4000
5000

In [13]:

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Assuming df is your DataFrame containing symptoms and prognosis
selected_features = ['itching', 'skin_rash', 'nodal_skin_eruptions', 'continuous_sneezing', 'shiv
                     'joint_pain', 'stomach_pain', 'acidity', 'ulcers_on_tongue', 'scurring', 's
                     'silver_like_dusting', 'small_dents_in_nails', 'inflammatory_nails', 'blist
                     'red_sore_around_nose', 'yellow_crust_ooze']

# Assuming your DataFrame is named 'train'
df_subset = train[selected_features]

# Convert non-numeric columns to numeric (if needed) or drop them
df_subset = df_subset.apply(pd.to_numeric, errors='coerce').dropna()

# Calculate the correlation matrix
corr_subset = df_subset.corr()

# Increase the size of the heatmap
plt.figure(figsize=(12, 10))

# Use Seaborn's heatmap with a coolwarm color map
sns.heatmap(corr_subset, annot=True, cmap='coolwarm', linewidths=.5, fmt=".2f",
            xticklabels=corr_subset.columns.values, yticklabels=corr_subset.columns.values)

plt.title('Correlation Matrix for Selected Features')
plt.show()
```


Correlation Matrix for Selected Features
itching 1.00 0.32 0.33 -0.09 -0.06 -0.18 -0.16 0.20 -0.09 -0.06 -0.06 -0.06 -0.06 -0.06 -0.06 -0.06 -0.06 -0.06
skin_rash 0.32 1.00 0.30 -0.09 -0.07 -0.03 0.17 0.16 -0.09 -0.07 0.32 0.33 0.33 0.33 0.33 0.33 0.33 0.33
nodal_skin_eruptions 0.33 0.30 1.00 -0.03 -0.02 -0.07 -0.06 -0.03 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02
continuous_sneezing -0.09 -0.09 -0.03 1.00 0.61 0.45 -0.09 -0.05 -0.05 -0.03 -0.03 -0.03 -0.03 -0.03 -0.03 -0.03 -0.03 -0.03
shivering -0.06 -0.07 -0.02 0.61 1.00 0.30 -0.06 -0.03 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02
chills -0.18 -0.03 -0.07 0.45 0.30 1.00 -0.00 -0.10 -0.10 -0.07 -0.07 -0.07 -0.07 -0.07 -0.07 -0.07 -0.07 -0.07
joint_pain -0.16 0.17 -0.06 -0.09 -0.06 -0.00 1.00 -0.09 -0.09 -0.06 -0.06 0.36 0.36 0.36 0.36 -0.06 -0.06 -0.06
stomach_pain 0.20 0.16 -0.03 -0.05 -0.03 -0.10 -0.09 1.00 0.43 0.65 -0.03 -0.03 -0.03 -0.03 -0.03 -0.03 -0.03 -0.03
acidity -0.09 -0.09 -0.03 -0.05 -0.03 -0.10 -0.09 0.43 1.00 0.61 -0.03 -0.03 -0.03 -0.03 -0.03 -0.03 -0.03 -0.03
ulcers_on_tongue -0.06 -0.07 -0.02 -0.03 -0.02 -0.07 -0.06 0.65 0.61 1.00 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02
scurring -0.06 0.32 -0.02 -0.03 -0.02 -0.07 -0.06 -0.03 -0.03 -0.02 1.00 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02
skin_peeling -0.06 0.33 -0.02 -0.03 -0.02 -0.07 0.36 -0.03 -0.03 -0.02 -0.02 1.00 0.95 0.95 0.95 -0.02 -0.02 -0.02
silver_like_dusting -0.06 0.33 -0.02 -0.03 -0.02 -0.07 0.36 -0.03 -0.03 -0.02 -0.02 0.95 1.00 0.95 0.95 -0.02 -0.02 -0.02
small_dents_in_nails -0.06 0.33 -0.02 -0.03 -0.02 -0.07 0.36 -0.03 -0.03 -0.02 -0.02 0.95 0.95 1.00 0.95 -0.02 -0.02 -0.02
inflammatory_nails -0.06 0.33 -0.02 -0.03 -0.02 -0.07 0.36 -0.03 -0.03 -0.02 -0.02 0.95 0.95 0.95 1.00 -0.02 -0.02 -0.02
blister -0.06 0.33 -0.02 -0.03 -0.02 -0.07 -0.06 -0.03 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 1.00 0.95 0.95
red_sore_around_nose -0.06 0.33 -0.02 -0.03 -0.02 -0.07 -0.06 -0.03 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 0.95 1.00 0.95
yellow_crust_ooze -0.06 0.33 -0.02 -0.03 -0.02 -0.07 -0.06 -0.03 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 0.95 0.95 1.00
hing rash ions zing ering chills pain pain idity ugue rring eling sting nails nails ister nose ooze
1.0 0.8 0.6 0.4 0.2 0.0

itc

skin_

nodal_skin_erupt

continuous_snee

shive

c

joint_

stomach_

ac

ulcers_on_ton

scu

skin_pee

silver_like_dus

small_dents_in_

inflammatory_

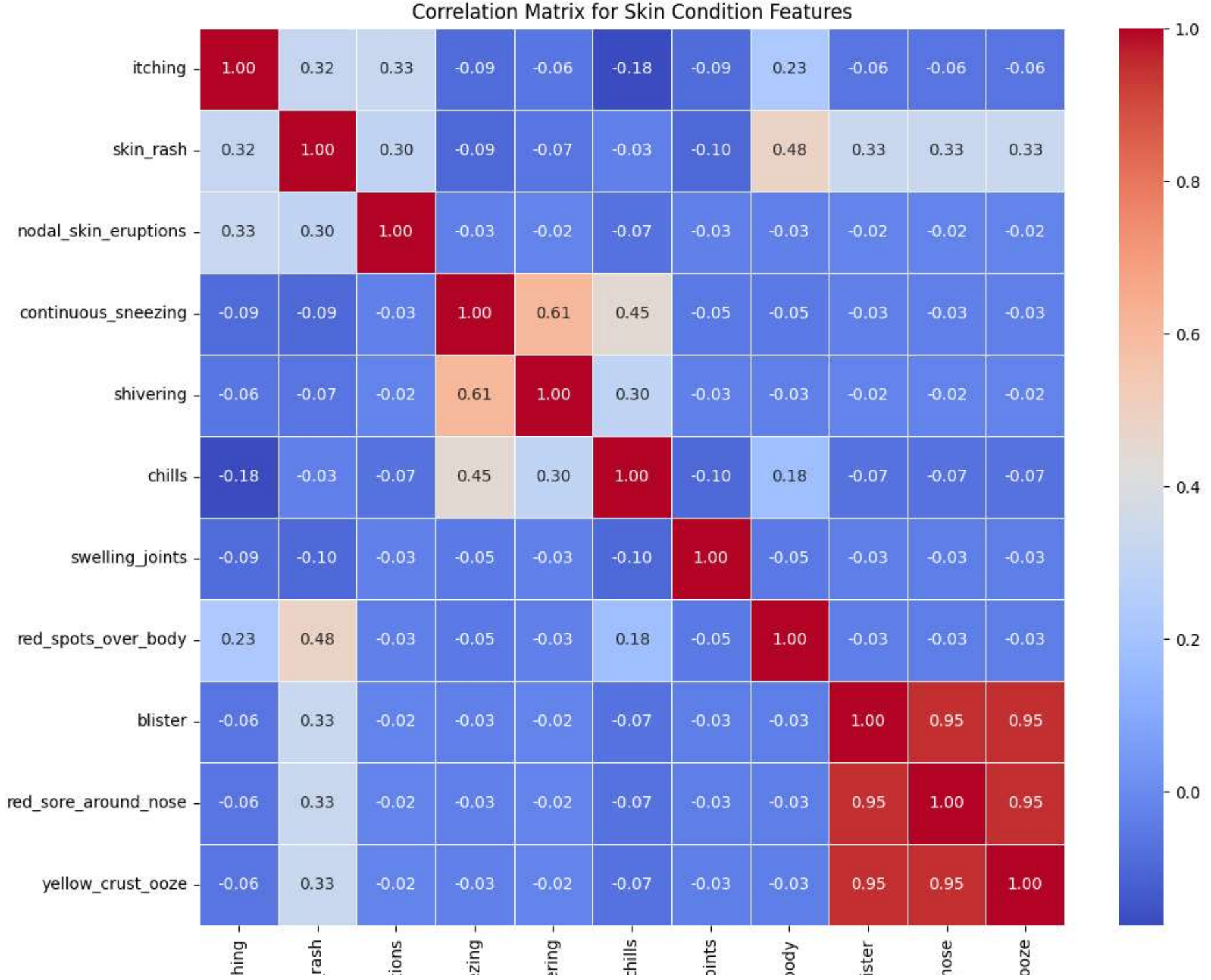
bli

red_sore_around_

yellow_crust_

In [14]:

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Assuming df is your DataFrame containing symptoms and prognosis
skin_condition_features = [
    'itching', 'skin_rash', 'nodal_skin_eruptions', 'continuous_sneezing', 'shivering', 'chills',
    'swelling_joints', 'red_spots_over_body', 'blister', 'red_sore_around_nose', 'yellow_crust_oo
]

# Assuming your DataFrame is named 'train'
df_skin_condition = train[skin_condition_features]

# Convert non-numeric columns to numeric (if needed) or drop them
df_skin_condition = df_skin_condition.apply(pd.to_numeric, errors='coerce').dropna()

# Calculate the correlation matrix
corr_skin_condition = df_skin_condition.corr()

# Increase the size of the heatmap
plt.figure(figsize=(12, 10))

# Use Seaborn's heatmap with a coolwarm color map
sns.heatmap(corr_skin_condition, annot=True, cmap='coolwarm', linewidths=.5, fmt=".2f",
            xticklabels=corr_skin_condition.columns.values, yticklabels=corr_skin_condition.colum

plt.title('Correlation Matrix for Skin Condition Features')
plt.show()
```


Correlation Matrix for Skin Condition Features
itching 1.00 0.32 0.33 -0.09 -0.06 -0.18 -0.09 0.23 -0.06 -0.06 -0.06
skin_rash 0.32 1.00 0.30 -0.09 -0.07 -0.03 -0.10 0.48 0.33 0.33 0.33
nodal_skin_eruptions 0.33 0.30 1.00 -0.03 -0.02 -0.07 -0.03 -0.03 -0.02 -0.02 -0.02
continuous_sneezing -0.09 -0.09 -0.03 1.00 0.61 0.45 -0.05 -0.05 -0.03 -0.03 -0.03
shivering -0.06 -0.07 -0.02 0.61 1.00 0.30 -0.03 -0.03 -0.02 -0.02 -0.02
chills -0.18 -0.03 -0.07 0.45 0.30 1.00 -0.10 0.18 -0.07 -0.07 -0.07
swelling_joints -0.09 -0.10 -0.03 -0.05 -0.03 -0.10 1.00 -0.05 -0.03 -0.03 -0.03
red_spots_over_body 0.23 0.48 -0.03 -0.05 -0.03 0.18 -0.05 1.00 -0.03 -0.03 -0.03
blister -0.06 0.33 -0.02 -0.03 -0.02 -0.07 -0.03 -0.03 1.00 0.95 0.95
red_sore_around_nose -0.06 0.33 -0.02 -0.03 -0.02 -0.07 -0.03 -0.03 0.95 1.00 0.95
yellow_crust_ooze -0.06 0.33 -0.02 -0.03 -0.02 -0.07 -0.03 -0.03 0.95 0.95 1.00
hing
rash
ions
zing
ering
chills
oints
body
ister
nose
ooze
1.0
0.8
0.6
0.4
0.2
0.0

itc

skin_

nodal_skin_erupt

continuous_snee

shive

c

swelling_j

red_spots_over_b

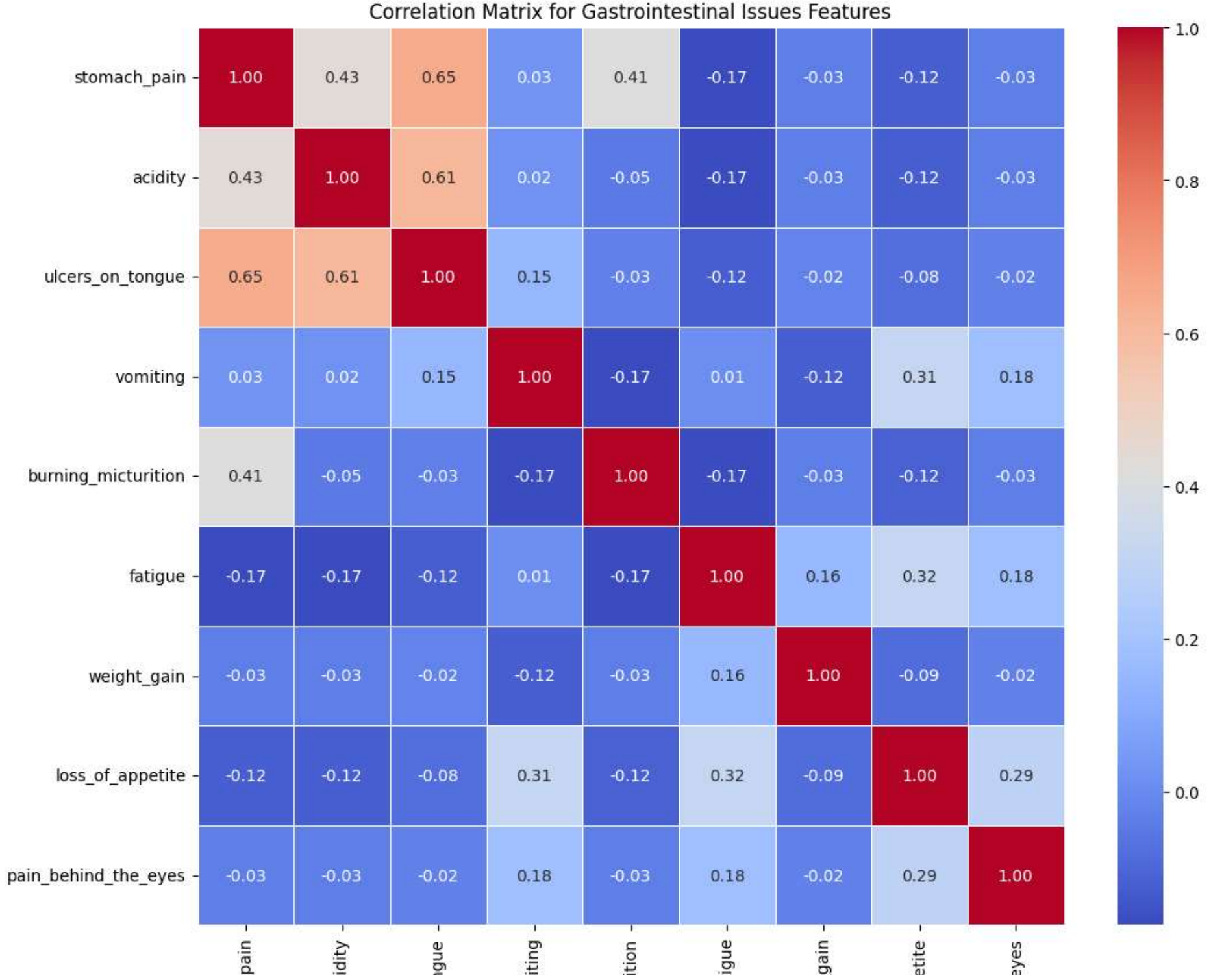
bli

red_sore_around_

yellow_crust_

In [15]:

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Assuming df is your DataFrame containing symptoms and prognosis
gi_issues_features = [
    'stomach_pain', 'acidity', 'ulcers_on_tongue', 'vomiting', 'burning_micturition',
    'fatigue', 'weight_gain', 'loss_of_appetite', 'pain_behind_the_eyes'
]

# Assuming your DataFrame is named 'train'
df_gi_issues = train[gi_issues_features]

# Convert non-numeric columns to numeric (if needed) or drop them
df_gi_issues = df_gi_issues.apply(pd.to_numeric, errors='coerce').dropna()

# Calculate the correlation matrix
corr_gi_issues = df_gi_issues.corr()

# Increase the size of the heatmap
plt.figure(figsize=(12, 10))

# Use Seaborn's heatmap with a coolwarm color map
sns.heatmap(corr_gi_issues, annot=True, cmap='coolwarm', linewidths=.5, fmt=".2f",
            xticklabels=corr_gi_issues.columns.values, yticklabels=corr_gi_issues.columns.values)

plt.title('Correlation Matrix for Gastrointestinal Issues Features')
plt.show()
```


Correlation Matrix for Gastrointestinal Issues Features
stomach_pain
acidity
ulcers_on_tongue
vomiting
burning_micturition
fatigue
weight_gain
loss_of_appetite
pain_behind_the_eyes
1.00 0.43 0.65 0.03 0.41 -0.17 -0.03 -0.12 -0.03
0.43 1.00 0.61 0.02 -0.05 -0.17 -0.03 -0.12 -0.03
0.65 0.61 1.00 0.15 -0.03 -0.12 -0.02 -0.08 -0.02
0.03 0.02 0.15 1.00 -0.17 0.01 -0.12 0.31 0.18
0.41 -0.05 -0.03 -0.17 1.00 -0.17 -0.03 -0.12 -0.03
-0.17 -0.17 -0.12 0.01 -0.17 1.00 0.16 0.32 0.18
-0.03 -0.03 -0.02 -0.12 -0.03 0.16 1.00 -0.09 -0.02
-0.12 -0.12 -0.08 0.31 -0.12 0.32 -0.09 1.00 0.29
-0.03 -0.03 -0.02 0.18 -0.03 0.18 -0.02 0.29 1.00
pain
idity
igue
iting
ition
igue
gain
etite
eyes
1.0
0.8
0.6
0.4
0.2
0.0


stomach_
aci
ulcers_on_ton
vomi
burning_micturi
fat
weight_
loss_of_appe
pain_behind_the_e

In [16]:
```
print(train.columns.tolist())
```

```
['itching', 'skin_rash', 'nodal_skin_eruptions', 'continuous_sneezing', 'shivering', 'chills',
'joint_pain', 'stomach_pain', 'acidity', 'ulcers_on_tongue', 'muscle_wasting', 'vomiting', 'burn
ing_micturition', 'spotting_ urination', 'fatigue', 'weight_gain', 'anxiety', 'cold_hands_and_fe
ets', 'mood_swings', 'weight_loss', 'restlessness', 'lethargy', 'patches_in_throat', 'irregular_
sugar_level', 'cough', 'high_fever', 'sunken_eyes', 'breathlessness', 'sweating', 'dehydration',
'indigestion', 'headache', 'yellowish_skin', 'dark_urine', 'nausea', 'loss_of_appetite', 'pain_b
ehind_the_eyes', 'back_pain', 'constipation', 'abdominal_pain', 'diarrhoea', 'mild_fever', 'yell
ow_urine', 'yellowing_of_eyes', 'acute_liver_failure', 'fluid_overload', 'swelling_of_stomach',
'swelled_lymph_nodes', 'malaise', 'blurred_and_distorted_vision', 'phlegm', 'throat_irritation',
'redness_of_eyes', 'sinus_pressure', 'runny_nose', 'congestion', 'chest_pain', 'weakness_in_limb
s', 'fast_heart_rate', 'pain_during_bowel_movements', 'pain_in_anal_region', 'bloody_stool', 'ir
ritation_in_anus', 'neck_pain', 'dizziness', 'cramps', 'bruising', 'obesity', 'swollen_legs', 's
wollen_blood_vessels', 'puffy_face_and_eyes', 'enlarged_thyroid', 'brittle_nails', 'swollen_extr
emeties', 'excessive_hunger', 'extra_marital_contacts', 'drying_and_tingling_lips', 'slurred_spe
ech', 'knee_pain', 'hip_joint_pain', 'muscle_weakness', 'stiff_neck', 'swelling_joints', 'moveme
nt_stiffness', 'spinning_movements', 'loss_of_balance', 'unsteadiness', 'weakness_of_one_body_si
de', 'loss_of_smell', 'bladder_discomfort', 'foul_smell_of urine', 'continuous_feel_of_urine',
'passage_of_gases', 'internal_itching', 'toxic_look_(typhos)', 'depression', 'irritability', 'mu
scle_pain', 'altered_sensorium', 'red_spots_over_body', 'belly_pain', 'abnormal_menstruation',
'dischromic _patches', 'watering_from_eyes', 'increased_appetite', 'polyuria', 'family_history',
'mucoid_sputum', 'rusty_sputum', 'lack_of_concentration', 'visual_disturbances', 'receiving_bloo
d_transfusion', 'receiving_unsterile_injections', 'coma', 'stomach_bleeding', 'distention_of_abd
omen', 'history_of_alcohol_consumption', 'fluid_overload.1', 'blood_in_sputum', 'prominent_veins
_on_calf', 'palpitations', 'painful_walking', 'pus_filled_pimples', 'blackheads', 'scurring', 's
kin_peeling', 'silver_like_dusting', 'small_dents_in_nails', 'inflammatory_nails', 'blister', 'r
ed_sore_around_nose', 'yellow_crust_ooze', 'prognosis']
```

In [17]:

```
import numpy as np

correlation_threshold = 0.9

numeric_df = train.drop('prognosis', axis=1)
corr_matrix = numeric_df.corr().abs()
upper_triangle = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(bool))
to_drop = [column for column in upper_triangle.columns if any(upper_triangle[column] > correlatio


df_filtered = numeric_df.drop(to_drop, axis=1)
```

In [18]:
```
print("Dropped columns With high correlation:")
print(to_drop)

print("\n\nFiltered columns:")
print(df_filtered.columns.tolist())
```

```
Dropped columns With high correlation:
['cold_hands_and_feets', 'redness_of_eyes', 'sinus_pressure', 'runny_nose', 'congestion', 'pain_
in_anal_region', 'bloody_stool', 'irritation_in_anus', 'bruising', 'swollen_legs', 'swollen_bloo
d_vessels', 'puffy_face_and_eyes', 'enlarged_thyroid', 'brittle_nails', 'swollen_extremeties',
'drying_and_tingling_lips', 'slurred_speech', 'hip_joint_pain', 'unsteadiness', 'loss_of_smell',
'continuous_feel_of_urine', 'internal_itching', 'altered_sensorium', 'belly_pain', 'abnormal_men
struation', 'increased_appetite', 'polyuria', 'receiving_blood_transfusion', 'receiving_unsteril
e_injections', 'coma', 'stomach_bleeding', 'distention_of_abdomen', 'history_of_alcohol_consumpt
ion', 'fluid_overload.1', 'prominent_veins_on_calf', 'palpitations', 'painful_walking', 'silver_
like_dusting', 'small_dents_in_nails', 'inflammatory_nails', 'red_sore_around_nose', 'yellow_cru
st_ooze']


Filtered columns:
['itching', 'skin_rash', 'nodal_skin_eruptions', 'continuous_sneezing', 'shivering', 'chills',
'joint_pain', 'stomach_pain', 'acidity', 'ulcers_on_tongue', 'muscle_wasting', 'vomiting', 'burn
ing_micturition', 'spotting_ urination', 'fatigue', 'weight_gain', 'anxiety', 'mood_swings', 'we
ight_loss', 'restlessness', 'lethargy', 'patches_in_throat', 'irregular_sugar_level', 'cough',
'high_fever', 'sunken_eyes', 'breathlessness', 'sweating', 'dehydration', 'indigestion', 'headac
he', 'yellowish_skin', 'dark_urine', 'nausea', 'loss_of_appetite', 'pain_behind_the_eyes', 'back
_pain', 'constipation', 'abdominal_pain', 'diarrhoea', 'mild_fever', 'yellow_urine', 'yellowing_
of_eyes', 'acute_liver_failure', 'fluid_overload', 'swelling_of_stomach', 'swelled_lymph_nodes',
'malaise', 'blurred_and_distorted_vision', 'phlegm', 'throat_irritation', 'chest_pain', 'weaknes
s_in_limbs', 'fast_heart_rate', 'pain_during_bowel_movements', 'neck_pain', 'dizziness', 'cramp
s', 'obesity', 'excessive_hunger', 'extra_marital_contacts', 'knee_pain', 'muscle_weakness', 'st
iff_neck', 'swelling_joints', 'movement_stiffness', 'spinning_movements', 'loss_of_balance', 'we
akness_of_one_body_side', 'bladder_discomfort', 'foul_smell_of urine', 'passage_of_gases', 'toxi
c_look_(typhos)', 'depression', 'irritability', 'muscle_pain', 'red_spots_over_body', 'dischromi
c _patches', 'watering_from_eyes', 'family_history', 'mucoid_sputum', 'rusty_sputum', 'lack_of_c
oncentration', 'visual_disturbances', 'blood_in_sputum', 'pus_filled_pimples', 'blackheads', 'sc
urring', 'skin_peeling', 'blister']
```

```
# import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Assuming df is your DataFrame containing features and target variable
selected_prognosis = 'Tuberculosis'
selected_columns = ['weight_loss', 'fatigue', 'prognosis']

# Filter the DataFrame for the selected prognosis
df_tuberculosis = train[train['prognosis'] == selected_prognosis]

# Melt the DataFrame for better visualization
df_melted = pd.melt(df_tuberculosis[selected_columns], id_vars='prognosis', var_name='variable',
value_name='value')

# Create a swarmplot
plt.figure(figsize=(10, 6))
sns.swarmplot(data=df_melted, x='variable', y='value', hue='prognosis', palette='Set2')
plt.title(f'Swarmplot for {selected_prognosis}')
plt.show()
```

In [19]:
```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42 entries, 0 to 41
Columns: 133 entries, itching to prognosis
dtypes: int64(132), object(1)
memory usage: 43.8+ KB
```

In [20]:
```
def preprocess_test_data(test, to_drop):
    test.drop(columns=to_drop, inplace=True)
    return test
```

In [21]:
```
test_data = preprocess_test_data(test, to_drop)
```

In [22]:
```
X = df_filtered
y = train['prognosis']
```

In [23]:
```
X_test = test_data.drop('prognosis', axis=1)
y_test = test_data['prognosis']

X_train, X_val, y_train, y_val = train_test_split(X, y, train_size =0.8, random_state=42)
```

In [24]:
```
print("Training set - X:", X_train.shape, "y:", y_train.shape)
print("Validation set - X:", X_val.shape, "y:", y_val.shape)
print("Testing set - X:", X_test.shape, "y:", y_test.shape)
```

```
Training set - X: (3936, 90) y: (3936,)
Validation set - X: (984, 90) y: (984,)
Testing set - X: (42, 90) y: (42,)
```

In [25]:
```
def evaluate_model(classifier):
    classifier.fit(X_train , y_train)
    y_pred = classifier.predict(X_val)
    yt_pred = classifier.predict(X_train)
    y_pred1 = classifier.predict(X_test)
    print('Training Accuracy: ', accuracy_score(y_train, yt_pred))
    print('Validation Accuracy: ', accuracy_score(y_val, y_pred))
    print('Testing Accuracy: ', accuracy_score(y_test, y_pred1))
    return [(accuracy_score(y_train, yt_pred)), (accuracy_score(y_val, y_pred)), (accuracy_score(
```

In [26]:
```
KNN = KNeighborsClassifier(n_neighbors=10)
KNN_result = evaluate_model(KNN)
```

```
Training Accuracy:  1.0
Validation Accuracy:  1.0
Testing Accuracy:  0.9761904761904762
```

In [27]:
```
SVM = SVC(C=1)
SVM_result = evaluate_model(SVM)
```

```
Training Accuracy:  1.0
Validation Accuracy:  1.0
Testing Accuracy:  0.9761904761904762
```

In [28]:
```
DTC = DecisionTreeClassifier(max_features=10)
DTC_result = evaluate_model(DTC)
```

```
Training Accuracy:  1.0
Validation Accuracy:  1.0
Testing Accuracy:  0.9761904761904762
```

In [29]:
```
RFC = RandomForestClassifier(max_depth = 13)
RFC_result = evaluate_model(RFC)
```

```
Training Accuracy:  1.0
Validation Accuracy:  1.0
Testing Accuracy:  0.9761904761904762
```

In [30]:
```
RFC_feature_importances = pd.DataFrame(RFC.feature_importances_,
                                       index=X_train.columns,
                                       columns=['Importance']).sort_values('Importance', ascending=Fa

RFC_feature_importances.head(50)
```

Out[30]:

	Importance
muscle_pain	0.024669
nausea	0.022081
throat_irritation	0.021463
weight_loss	0.020420
passage_of_gases	0.018745
skin_peeling	0.018095
blister	0.017384
high_fever	0.017196
swelling_of_stomach	0.017024
fast_heart_rate	0.016756
muscle_weakness	0.016702
fatigue	0.016547
red_spots_over_body	0.016098
movement_stiffness	0.016056
malaise	0.015906
headache	0.015902
yellowing_of_eyes	0.015683
abdominal_pain	0.015464
mild_fever	0.014559
depression	0.014495
knee_pain	0.014428

	Importance
swelled_lymph_nodes	0.014341
loss_of_appetite	0.014335
phlegm	0.014165
blood_in_sputum	0.014004
irritability	0.013832
itching	0.013777
spinning_movements	0.013742
irregular_sugar_level	0.013596
weight_gain	0.013313
dark_urine	0.013300
acute_liver_failure	0.012899
lethargy	0.012833
dischromic _patches	0.012768
excessive_hunger	0.012706
back_pain	0.012638
obesity	0.012616
swelling_joints	0.012613
loss_of_balance	0.012578
weakness_of_one_body_side	0.012482
neck_pain	0.012442
joint_pain	0.012396
lack_of_concentration	0.012271

	Importance
indigestion	0.011878
toxic_look_(typhos)	0.011847
chills	0.011727
pain_behind_the_eyes	0.011651
sweating	0.011632
constipation	0.011545
restlessness	0.011089

In [31]:
```
print("Random Forest Classifier Model\n")
for num_features in range(1, 91, 10):
    # Select top N features
    top_features = RFC_feature_importances.head(num_features).index
    X_train_selected = X_train[top_features]
    X_val_selected = X_val[top_features]

    # Train the model
    RFC_selected = RandomForestClassifier()
    RFC_selected.fit(X_train_selected, y_train)

    # Evaluate accuracy on both training and validation sets
    train_accuracy = RFC_selected.score(X_train_selected, y_train)
    val_accuracy = RFC_selected.score(X_val_selected, y_val)

    print(f"Accuracy with {num_features} features - Training Accuracy: {train_accuracy} , Validat
```

Random Forest Classifier Model

Accuracy with 1 features - Training Accuracy: 0.051321138211382115 , Validation Accuracy: 0.0386
1788617886179
Accuracy with 11 features - Training Accuracy: 0.3821138211382114 , Validation Accuracy: 0.34959
349593495936
Accuracy with 21 features - Training Accuracy: 0.7345020325203252 , Validation Accuracy: 0.71443
08943089431
Accuracy with 31 features - Training Accuracy: 0.7688008130081301 , Validation Accuracy: 0.74796
74796747967
Accuracy with 41 features - Training Accuracy: 0.845020325203252 , Validation Accuracy: 0.827235
7723577236
Accuracy with 51 features - Training Accuracy: 0.9138719512195121 , Validation Accuracy: 0.89939
0243902439
Accuracy with 61 features - Training Accuracy: 0.9903455284552846 , Validation Accuracy: 0.98983
73983739838
Accuracy with 71 features - Training Accuracy: 0.9939024390243902 , Validation Accuracy: 0.99390
24390243902
Accuracy with 81 features - Training Accuracy: 1.0 , Validation Accuracy: 1.0

In [32]:
```
print("KNN Model\n")
for num_features in range(1, 91, 10):
    # Select top N features
    top_features = RFC_feature_importances.head(num_features).index
    X_train_selected = X_train[top_features]
    X_val_selected = X_val[top_features]

    # Train the model
    KNN_selected = KNeighborsClassifier()
    KNN_selected.fit(X_train_selected, y_train)

    # Evaluate accuracy on both training and validation sets
    train_accuracy = KNN_selected.score(X_train_selected, y_train)
    val_accuracy = KNN_selected.score(X_val_selected, y_val)
    print(f"Accuracy with {num_features} features - Training Accuracy: {train_accuracy} , Validat
```

KNN Model

Accuracy with 1 features - Training Accuracy: 0.049796747967479675 , Validation Accuracy: 0.0447154471545
Accuracy with 11 features - Training Accuracy: 0.3788109756097561 , Validation Accuracy: 0.3567073170731707
Accuracy with 21 features - Training Accuracy: 0.7322154471544715 , Validation Accuracy: 0.7235772357723578
Accuracy with 31 features - Training Accuracy: 0.7680386178861789 , Validation Accuracy: 0.7571138211382114
Accuracy with 41 features - Training Accuracy: 0.8442581300813008 , Validation Accuracy: 0.8241869918699187
Accuracy with 51 features - Training Accuracy: 0.9128556910569106 , Validation Accuracy: 0.9034552845528455
Accuracy with 61 features - Training Accuracy: 0.9890752032520326 , Validation Accuracy: 0.9888211382113821
Accuracy with 71 features - Training Accuracy: 0.993140243902439 , Validation Accuracy: 0.9908536585365854
Accuracy with 81 features - Training Accuracy: 0.9994918699186992 , Validation Accuracy: 0.9959349593495935

In [33]:
```
# Choosing the top 50 features as there is little change in accuracy from 50 to 80 features

top_features = RFC_feature_importances.head(50).index
X1 = X[top_features]
y1 = y
X_train1, X_val1, y_train1, y_val1 = train_test_split(X1, y1, train_size=0.8, random_state=42)
X_test1 = X_test[top_features]
X1.columns
```

Out[33]:
```
Index(['muscle_pain', 'nausea', 'throat_irritation', 'weight_loss',
       'passage_of_gases', 'skin_peeling', 'blister', 'high_fever',
       'swelling_of_stomach', 'fast_heart_rate', 'muscle_weakness', 'fatigue',
       'red_spots_over_body', 'movement_stiffness', 'malaise', 'headache',
       'yellowing_of_eyes', 'abdominal_pain', 'mild_fever', 'depression',
       'knee_pain', 'swelled_lymph_nodes', 'loss_of_appetite', 'phlegm',
       'blood_in_sputum', 'irritability', 'itching', 'spinning_movements',
       'irregular_sugar_level', 'weight_gain', 'dark_urine',
       'acute_liver_failure', 'lethargy', 'dischromic _patches',
       'excessive_hunger', 'back_pain', 'obesity', 'swelling_joints',
       'loss_of_balance', 'weakness_of_one_body_side', 'neck_pain',
       'joint_pain', 'lack_of_concentration', 'indigestion',
       'toxic_look_(typhos)', 'chills', 'pain_behind_the_eyes', 'sweating',
       'constipation', 'restlessness'],
      dtype='object')
```

In [34]:
```
# Adding serial numbers to the list of columns
columns_with_serial = list(enumerate(X1.columns, start=1))

# Displaying the serial numbers and corresponding column names
for serial, column in columns_with_serial:
    print(f"{serial}. {column}")
```

1. muscle_pain
2. nausea
3. throat_irritation
4. weight_loss
5. passage_of_gases
6. skin_peeling
7. blister
8. high_fever
9. swelling_of_stomach
10. fast_heart_rate
11. muscle_weakness
12. fatigue
13. red_spots_over_body
14. movement_stiffness
15. malaise
16. headache
17. yellowing_of_eyes
18. abdominal_pain
19. mild_fever
20. depression
21. knee_pain
22. swelled_lymph_nodes
23. loss_of_appetite
24. phlegm
25. blood_in_sputum
26. irritability
27. itching
28. spinning_movements
29. irregular_sugar_level
30. weight_gain
31. dark_urine
32. acute_liver_failure
33. lethargy
34. dischromic _patches

35. excessive_hunger
36. back_pain
37. obesity
38. swelling_joints
39. loss_of_balance
40. weakness_of_one_body_side
41. neck_pain
42. joint_pain
43. lack_of_concentration
44. indigestion
45. toxic_look_(typhos)
46. chills
47. pain_behind_the_eyes
48. sweating
49. constipation
50. restlessness

In [35]:
```
RFC_model = RandomForestClassifier()
RFC_model.fit(X_train1, y_train1)
y_train_pred1 = RFC_model.predict(X_train1)
y_val_pred1 = RFC_model.predict(X_val1)
y_test_pred1 = RFC_model.predict(X_test1)

print('Training Accuracy: ', accuracy_score(y_train1, y_train_pred1))
print('Validation Accuracy: ', accuracy_score(y_val1, y_val_pred1))
print('Testing Accuracy: ', accuracy_score(y_test, y_test_pred1))
```

```
Training Accuracy:  0.9138719512195121
Validation Accuracy:  0.899390243902439
Testing Accuracy:  0.9047619047619048
```

In [36]:
```
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train1, y_train1)
y_train_pred1 = knn_model.predict(X_train1)
y_val_pred1 = knn_model.predict(X_val1)
y_test_pred1 = knn_model.predict(X_test1)

print('Training Accuracy: ', accuracy_score(y_train1, y_train_pred1))
print('Validation Accuracy: ', accuracy_score(y_val1, y_val_pred1))
print('Testing Accuracy: ', accuracy_score(y_test, y_test_pred1))
```

```
Training Accuracy:  0.9128556910569106
Validation Accuracy:  0.9034552845528455
Testing Accuracy:  0.9047619047619048
```

In [37]:
```
test_predictions = pd.DataFrame(y_test_pred1, columns=["predicted"])
result_df = test.join(test_predictions)[["prognosis", "predicted"]]
result_df.head(10)
```

Out[37]:

	prognosis	predicted
0	Fungal infection	Fungal infection
1	Allergy	Allergy
2	GERD	GERD
3	Chronic cholestasis	Chronic cholestasis
4	Drug Reaction	Drug Reaction
5	Peptic ulcer diseae	Peptic ulcer diseae
6	AIDS	AIDS
7	Diabetes	Diabetes
8	Gastroenteritis	GERD
9	Bronchial Asthma	Bronchial Asthma

In [38]:
```
pickle.dump(knn_model , open('knn_model.pkl' ,'wb'))
```

In [39]:
```
X1.columns
```

Out[39]:
```
Index(['muscle_pain', 'nausea', 'throat_irritation', 'weight_loss',
       'passage_of_gases', 'skin_peeling', 'blister', 'high_fever',
       'swelling_of_stomach', 'fast_heart_rate', 'muscle_weakness', 'fatigue',
       'red_spots_over_body', 'movement_stiffness', 'malaise', 'headache',
       'yellowing_of_eyes', 'abdominal_pain', 'mild_fever', 'depression',
       'knee_pain', 'swelled_lymph_nodes', 'loss_of_appetite', 'phlegm',
       'blood_in_sputum', 'irritability', 'itching', 'spinning_movements',
       'irregular_sugar_level', 'weight_gain', 'dark_urine',
       'acute_liver_failure', 'lethargy', 'dischromic _patches',
       'excessive_hunger', 'back_pain', 'obesity', 'swelling_joints',
       'loss_of_balance', 'weakness_of_one_body_side', 'neck_pain',
       'joint_pain', 'lack_of_concentration', 'indigestion',
       'toxic_look_(typhos)', 'chills', 'pain_behind_the_eyes', 'sweating',
       'constipation', 'restlessness'],
      dtype='object')
```