Pneumonia diagnosis using chest x-ray images

July 26, 2024

1 Data processing

[ ]:
```python
import cv2
import pandas as pd
import os
import warnings
import numpy as np

warnings.filterwarnings('ignore', message='Intel MKL WARNING')

def process_data(file_path, data_list):
    data = cv2.imread(file_path)
    if data is not None:
        data_resized = cv2.resize(data, (64, 64))
        data_flattened = data_resized.flatten()
        data_normalized = data_flattened / 255.0  # Normalize to [0, 1]
        return data_list.append(data_normalized)
```

[ ]:
```python
# Convert train normal into data

train_normal_data = []

# Iterate through files in the folder
for filename in os.listdir('../data/train/NORMAL'):
    file_path = os.path.join('../data/train/NORMAL', filename)
    if os.path.isfile(file_path):
        process_data(file_path, train_normal_data)

train_normal_df = pd.DataFrame(train_normal_data)
value_to_set = "disease"
columns = list(train_normal_df.columns)
columns = [value_to_set] + columns[:-1]
train_normal_df = train_normal_df.reindex(columns=columns)
fill_value = 0
train_normal_df.fillna(fill_value, inplace=True)
train_normal_df.head()
```

```
[ ]:   disease         0         1         2         3         4         5  \
     0      0.0  0.117647  0.117647  0.117647  0.109804  0.109804  0.109804
     1      0.0  0.231373  0.231373  0.231373  0.188235  0.188235  0.188235
     2      0.0  0.203922  0.203922  0.203922  0.286275  0.286275  0.286275
     3      0.0  0.007843  0.007843  0.007843  0.113725  0.113725  0.113725
     4      0.0  0.466667  0.466667  0.466667  0.486275  0.486275  0.486275

               6         7         8  …     12277     12278     12279     12280  \
     0  0.105882  0.105882  0.105882  …  0.000000  0.000000  0.000000  0.000000
     1  0.309804  0.309804  0.309804  …  0.290196  0.290196  0.125490  0.125490
     2  0.356863  0.356863  0.356863  …  0.000000  0.000000  0.000000  0.000000
     3  0.168627  0.168627  0.168627  …  0.000000  0.000000  0.000000  0.000000
     4  0.517647  0.517647  0.517647  …  0.254902  0.254902  0.137255  0.137255

           12281  12282  12283  12284  12285  12286
     0  0.000000    0.0    0.0    0.0    0.0    0.0
     1  0.125490    0.0    0.0    0.0    0.0    0.0
     2  0.000000    0.0    0.0    0.0    0.0    0.0
     3  0.000000    0.0    0.0    0.0    0.0    0.0
     4  0.137255    0.0    0.0    0.0    0.0    0.0

     [5 rows x 12288 columns]
```

```
[ ]: # Convert train pneumonia into data

     train_pneumonia_data = []

     # Iterate through files in the folder
     for filename in os.listdir('../data/train/PNEUMONIA'):
         file_path = os.path.join('../data/train/PNEUMONIA', filename)
         if os.path.isfile(file_path):
             process_data(file_path, train_pneumonia_data)

     train_pneumonia_df = pd.DataFrame(train_pneumonia_data)
     train_pneumonia_df = pd.DataFrame(train_pneumonia_df)
     value_to_set = "disease"
     columns = list(train_pneumonia_df.columns)
     columns = [value_to_set] + columns[:-1]
     train_pneumonia_df = train_pneumonia_df.reindex(columns=columns)
     fill_value = 1
     train_pneumonia_df.fillna(fill_value, inplace=True)
     train_pneumonia_df.head()
```

```
[ ]:   disease         0         1         2         3         4         5  \
     0      1.0  0.803922  0.803922  0.803922  0.792157  0.792157  0.792157
     1      1.0  0.039216  0.039216  0.039216  0.125490  0.125490  0.125490
     2      1.0  0.172549  0.172549  0.172549  0.172549  0.172549  0.172549
```

```
3      1.0  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
4      1.0  0.752941  0.752941  0.752941  0.521569  0.521569  0.521569

          6         7         8  …     12277     12278     12279     12280  \
0  0.811765  0.811765  0.811765  …  0.098039  0.098039  0.133333  0.133333
1  0.196078  0.196078  0.196078  …  0.000000  0.000000  0.000000  0.000000
2  0.160784  0.160784  0.160784  …  0.180392  0.180392  0.172549  0.172549
3  0.000000  0.000000  0.000000  …  0.000000  0.000000  0.000000  0.000000
4  0.239216  0.239216  0.239216  …  0.215686  0.215686  0.035294  0.035294

      12281     12282     12283     12284     12285     12286
0  0.133333  0.168627  0.168627  0.168627  0.184314  0.184314
1  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
2  0.172549  0.192157  0.192157  0.192157  0.211765  0.211765
3  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
4  0.035294  0.054902  0.054902  0.054902  0.058824  0.058824

[5 rows x 12288 columns]
```

[ ]:
```python
# Convert test normal into data

test_normal_data = []

# Iterate through files in the folder
for filename in os.listdir('../data/test/NORMAL'):
    file_path = os.path.join('../data/test/NORMAL', filename)
    if os.path.isfile(file_path):
        process_data(file_path, test_normal_data)

test_normal_df = pd.DataFrame(test_normal_data)
test_normal_df = pd.DataFrame(test_normal_df)
value_to_set = "disease"
columns = list(test_normal_df.columns)
columns = [value_to_set] + columns[:-1]
test_normal_df = test_normal_df.reindex(columns=columns)
fill_value = 0
test_normal_df.fillna(fill_value, inplace=True)
test_normal_df.head()
```

[ ]:
```
   disease         0         1         2         3         4         5  \
0      0.0  0.082353  0.082353  0.082353  0.086275  0.086275  0.086275
1      0.0  0.011765  0.011765  0.011765  0.011765  0.011765  0.011765
2      0.0  0.270588  0.270588  0.270588  0.215686  0.215686  0.215686
3      0.0  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
4      0.0  0.074510  0.074510  0.074510  0.290196  0.290196  0.290196

          6         7         8  …     12277     12278     12279     12280  \
0  0.074510  0.074510  0.074510  …  0.133333  0.133333  0.145098  0.145098
1  0.015686  0.015686  0.015686  …  0.031373  0.031373  0.023529  0.023529
2  0.200000  0.200000  0.200000  …  0.000000  0.000000  0.000000  0.000000
3  0.047059  0.047059  0.047059  …  0.000000  0.000000  0.000000  0.000000
4  0.372549  0.372549  0.372549  …  0.000000  0.000000  0.007843  0.007843

      12281     12282     12283     12284     12285     12286
0  0.145098  0.094118  0.094118  0.094118  0.160784  0.160784
1  0.023529  0.019608  0.019608  0.019608  0.019608  0.019608
2  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
3  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
4  0.007843  0.035294  0.035294  0.035294  0.023529  0.023529

[5 rows x 12288 columns]
```

[ ]:
```
# Convert test pneumonia into data

test_pneumonia_data = []

# Iterate through files in the folder
for filename in os.listdir('../data/test/PNEUMONIA'):
    file_path = os.path.join('../data/test/PNEUMONIA', filename)
    if os.path.isfile(file_path):
        process_data(file_path, test_pneumonia_data)

test_pneumonia_df = pd.DataFrame(test_pneumonia_data)
test_pneumonia_df = pd.DataFrame(test_pneumonia_df)
value_to_set = "disease"
columns = list(test_pneumonia_df.columns)
columns = [value_to_set] + columns[:-1]
test_pneumonia_df = test_pneumonia_df.reindex(columns=columns)
fill_value = 1
test_pneumonia_df.fillna(fill_value, inplace=True)
test_pneumonia_df.head()
```

[ ]:
```
   disease         0         1         2         3         4         5  \
0      1.0  0.105882  0.105882  0.105882  0.486275  0.486275  0.486275
1      1.0  0.513725  0.513725  0.513725  0.294118  0.294118  0.294118
2      1.0  0.847059  0.847059  0.847059  0.815686  0.815686  0.815686
3      1.0  0.462745  0.462745  0.462745  0.372549  0.372549  0.372549
4      1.0  0.090196  0.090196  0.090196  0.223529  0.223529  0.223529

          6         7         8  …     12277     12278     12279     12280  \
0  0.137255  0.137255  0.137255  …  0.086275  0.086275  0.090196  0.090196
1  0.015686  0.015686  0.015686  …  0.058824  0.058824  0.066667  0.066667
2  0.858824  0.858824  0.858824  …  0.101961  0.101961  0.027451  0.027451
3  0.298039  0.298039  0.298039  …  0.054902  0.054902  0.074510  0.074510
```

```
4  0.313725  0.313725  0.313725  …  0.003922  0.003922  0.003922  0.003922

      12281     12282     12283     12284     12285     12286
0  0.090196  0.098039  0.098039  0.098039  0.098039  0.098039
1  0.066667  0.070588  0.070588  0.070588  0.074510  0.074510
2  0.027451  0.050980  0.050980  0.050980  0.058824  0.058824
3  0.074510  0.086275  0.086275  0.086275  0.090196  0.090196
4  0.003922  0.003922  0.003922  0.003922  0.003922  0.003922

[5 rows x 12288 columns]
```

[ ]:
```
print("Number of rows:", train_normal_df.shape[0])
print("Number of rows:", train_pneumonia_df.shape[0])
print("Number of rows:", test_normal_df.shape[0])
print("Number of rows:", test_pneumonia_df.shape[0])
```

```
Number of rows: 1349
Number of rows: 3883
Number of rows: 234
Number of rows: 390
```

[ ]:
```
df = pd.concat([train_normal_df, train_pneumonia_df, test_normal_df,␣
 ↪test_pneumonia_df], axis=0)
df.head()
```

[ ]:
```
   disease         0         1         2         3         4         5  \
0      0.0  0.117647  0.117647  0.117647  0.109804  0.109804  0.109804
1      0.0  0.231373  0.231373  0.231373  0.188235  0.188235  0.188235
2      0.0  0.203922  0.203922  0.203922  0.286275  0.286275  0.286275
3      0.0  0.007843  0.007843  0.007843  0.113725  0.113725  0.113725
4      0.0  0.466667  0.466667  0.466667  0.486275  0.486275  0.486275

          6         7         8  …     12277     12278     12279     12280  \
0  0.105882  0.105882  0.105882  …  0.000000  0.000000  0.000000  0.000000
1  0.309804  0.309804  0.309804  …  0.290196  0.290196  0.125490  0.125490
2  0.356863  0.356863  0.356863  …  0.000000  0.000000  0.000000  0.000000
3  0.168627  0.168627  0.168627  …  0.000000  0.000000  0.000000  0.000000
4  0.517647  0.517647  0.517647  …  0.254902  0.254902  0.137255  0.137255

      12281  12282  12283  12284  12285  12286
0  0.000000    0.0    0.0    0.0    0.0    0.0
1  0.125490    0.0    0.0    0.0    0.0    0.0
2  0.000000    0.0    0.0    0.0    0.0    0.0
3  0.000000    0.0    0.0    0.0    0.0    0.0
4  0.137255    0.0    0.0    0.0    0.0    0.0

[5 rows x 12288 columns]
```

2 Dimension reduction: singular value decomposition

[ ]:
```
X = df.drop(df.columns[0], axis=1)
y = df[df.columns[0]]
```

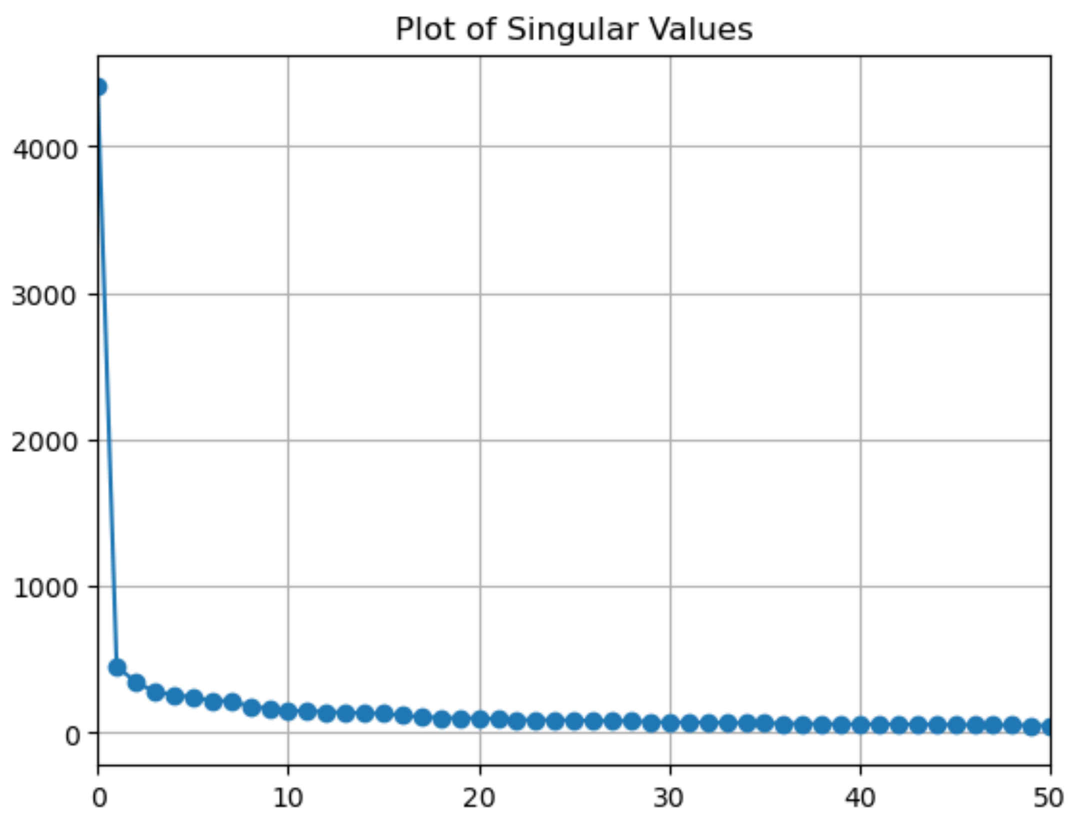
[ ]:
```
# Compute SVD
U, D, V = np.linalg.svd(X, full_matrices=False)
```

```
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel(R)
SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math Kernel
Library 2025.0 will require Intel(R) Advanced Vector Extensions (Intel(R) AVX)
instructions.
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel(R)
SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math Kernel
Library 2025.0 will require Intel(R) Advanced Vector Extensions (Intel(R) AVX)
instructions.
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel(R)
SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math Kernel
Library 2025.0 will require Intel(R) Advanced Vector Extensions (Intel(R) AVX)
instructions.
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel(R)
SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math Kernel
Library 2025.0 will require Intel(R) Advanced Vector Extensions (Intel(R) AVX)
instructions.
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel(R)
SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math Kernel
Library 2025.0 will require Intel(R) Advanced Vector Extensions (Intel(R) AVX)
instructions.
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel(R)
SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math Kernel
Library 2025.0 will require Intel(R) Advanced Vector Extensions (Intel(R) AVX)
instructions.
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel(R)
SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math Kernel
Library 2025.0 will require Intel(R) Advanced Vector Extensions (Intel(R) AVX)
instructions.
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel(R)
SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math Kernel
Library 2025.0 will require Intel(R) Advanced Vector Extensions (Intel(R) AVX)
instructions.
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel(R)
SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math Kernel
Library 2025.0 will require Intel(R) Advanced Vector Extensions (Intel(R) AVX)
instructions.
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel(R)
SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math Kernel
Library 2025.0 will require Intel(R) Advanced Vector Extensions (Intel(R) AVX)
instructions.
```

```
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel(R)
SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math Kernel
Library 2025.0 will require Intel(R) Advanced Vector Extensions (Intel(R) AVX)
instructions.
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel(R)
SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math Kernel
Library 2025.0 will require Intel(R) Advanced Vector Extensions (Intel(R) AVX)
instructions.
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel(R)
SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math Kernel
Library 2025.0 will require Intel(R) Advanced Vector Extensions (Intel(R) AVX)
instructions.
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel(R)
SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math Kernel
Library 2025.0 will require Intel(R) Advanced Vector Extensions (Intel(R) AVX)
instructions.
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel(R)
SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math Kernel
Library 2025.0 will require Intel(R) Advanced Vector Extensions (Intel(R) AVX)
instructions.
```

[ ]:
```python
import matplotlib.pyplot as plt

x_values = np.arange(len(D))
plt.plot(x_values, D, marker='o', linestyle='-')
plt.xlim(0, 50)
plt.title('Plot of Singular Values')
plt.grid(True)
plt.show()
```



Based on this elbow plot, additional components beyond 20 would not contribute much additional information to the model.

```
[ ]: # Dimension-reduced X
     V20 = np.transpose(V)[:, :20]
     DR_X = X.values @ V20
```

```
[ ]: arr = y.values
     y = arr.reshape(-1, 1)
     df = pd.concat([pd.DataFrame(y).rename(columns={0: 'disease'}), pd.
      ↪DataFrame(DR_X)], axis=1)
     df.head()
```

```
[ ]:    disease          0          1         2         3         4         5  \
     0      0.0 -55.337547  11.189257  0.969288 -8.103200  2.430869 -1.146683
     1      0.0 -54.702970  -4.086767 -4.363943  3.081711 -0.664137  1.970735
     2      0.0 -63.044600   1.002935 -7.537691 -6.473655  1.696616  2.288390
     3      0.0 -54.321275  -5.898212 -2.937312 -8.790238  0.726498 -0.568197
     4      0.0 -67.651331   6.718537 -7.945233 -0.536995  1.810724  3.996124

               6          7          8  …         10         11         12         13  \
     0  2.988481 -4.039937  6.199420  …  0.981185 -0.509654  4.006090 -0.193792
     1  0.320779 -1.787443  2.583499  … -0.473178 -1.701906 -0.431524  1.228657
     2  1.674289 -0.141939 -1.281375  … -2.079441 -0.387920 -1.936197 -0.314772
     3  2.704220  4.149871 -0.841565  …  2.641541  1.463082 -0.473949 -0.956462
     4 -2.179091  0.839701 -2.102178  … -2.362217  1.960484  2.895017  0.562866

              14        15        16        17        18        19
     0  0.568191 -1.433567  0.050123 -0.263677  1.224021 -1.287958
     1 -1.162580  1.818841  1.238074  1.130522 -1.587421  1.303172
     2 -0.884854 -2.530839 -0.716002  0.965882 -1.142154 -0.603895
     3 -2.194333 -1.976931 -1.922920 -0.338298 -0.184530  0.363268
     4  2.162667  1.745250  0.131177  0.026476 -0.984771  1.237131

     [5 rows x 21 columns]
```

3 Testing classifiers, including further investigation of SVM kernels

[ ]:
```python
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import roc_auc_score
from sklearn.metrics import accuracy_score

classifiers = {
    'Logistic Regression': LogisticRegression(),
    'Random Forest': RandomForestClassifier(),
    'K Nearest Neighbors': KNeighborsClassifier(),
    'Support Vector Machine (RBF)': SVC(probability=True),
    'Support Vector Machine (Linear)': SVC(kernel = "linear", probability=True),
    'Support Vector Machine (Polynomial)': SVC(kernel = "poly",␣
 ↪probability=True),
    'Support Vector Machine (Sigmoid)': SVC(kernel = "sigmoid",␣
 ↪probability=True),
}

skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

scores = {name: {'Accuracy': [], 'AUC': []} for name in classifiers.keys()}

for train_index, test_index in skf.split(df.drop(columns='disease'),␣
 ↪df['disease']):
    X_train, X_test = df.drop(columns='disease').iloc[train_index], df.
↪drop(columns='disease').iloc[test_index]
    y_train, y_test = df['disease'].iloc[train_index], df['disease'].
↪iloc[test_index]

    for clf_name, clf in classifiers.items():
        # Train classifier
        clf.fit(X_train, y_train)

        # Make predictions
        y_pred = clf.predict(X_test)

        # Make predictions for AUC if possible, check if the classifier␣
↪supports predict_proba
        if hasattr(clf, "predict_proba"):
            y_prob = clf.predict_proba(X_test)[:, 1]
            auc = roc_auc_score(y_test, y_prob)
            scores[clf_name]['AUC'].append(auc)
        else:
            y_prob = clf.decision_function(X_test)  # For models like SVC with␣
↪'probability=False'
            auc = roc_auc_score(y_test, y_prob)
            scores[clf_name]['AUC'].append(auc)

        accuracy = accuracy_score(y_test, y_pred)
        scores[clf_name]['Accuracy'].append(accuracy)

print()
print()

for clf_name in classifiers:
    mean_accuracy = np.mean(scores[clf_name]['Accuracy'])
    mean_auc = np.mean(scores[clf_name]['AUC'])
    print(f"{clf_name}: Mean Accuracy = {mean_accuracy * 100:.2f}%, Mean AUC =␣
↪{mean_auc:.4f}")
```

```
/Users/brookestevens/opt/anaconda3/envs/myenv/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/Users/brookestevens/opt/anaconda3/envs/myenv/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/Users/brookestevens/opt/anaconda3/envs/myenv/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/Users/brookestevens/opt/anaconda3/envs/myenv/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/Users/brookestevens/opt/anaconda3/envs/myenv/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/Users/brookestevens/opt/anaconda3/envs/myenv/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/Users/brookestevens/opt/anaconda3/envs/myenv/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/Users/brookestevens/opt/anaconda3/envs/myenv/lib/python3.9/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(


Logistic Regression: Mean Accuracy = 92.64%, Mean AUC = 0.9718
Random Forest: Mean Accuracy = 93.02%, Mean AUC = 0.9731
K Nearest Neighbors: Mean Accuracy = 92.40%, Mean AUC = 0.9555
Support Vector Machine (RBF): Mean Accuracy = 93.07%, Mean AUC = 0.9756
Support Vector Machine (Linear): Mean Accuracy = 92.74%, Mean AUC = 0.9717
Support Vector Machine (Polynomial): Mean Accuracy = 92.57%, Mean AUC = 0.9704
Support Vector Machine (Sigmoid): Mean Accuracy = 78.69%, Mean AUC = 0.7981
```