Survey on Algorithms for Personalized Patient Health Informatics and Diagnosis

CS 6045
Advanced Algorithms

Spring 2025

Michael Rizig

Instructor – Dr. Michail Alexiou

I. Abstract

The goal of this research is to provide an analysis on the current algorithms used for detection and diagnosis of disease or illness. This research includes discussing the 4 primary approaches to the development of these type algorithms: Supervised Learning, Unsupervised Learning, Statistical Modeling, and custom-made algorithms that utilize some combination of both. We will discuss their strengths and weaknesses, and how they fit into the larger scheme of overall patient diagnosis. This research also covers the primary data sources for collecting anonymous patient data for developing and training machine learning classification algorithms, the use and considerations of this data, and results of each approach.

II. DATA COLLECTION

Collecting high quality data to train models on is the first step to any approach. There are several sources that one can use to collect data ethically. Here are 3 common sources:

MEDICAL IMAGING ARCHIVES (PACS): These archives contain anonymous medical images such as x-rays, scans, MRIS, and other digital medical images. This source would be useful for training a model in machine vision to detect anomalies or cancers in a radiology environment.
EXAMPLES:
https://www.cancerimagingarchive.net/
https://grand-challenge.org/
https://www.cancerimagingarchive.net/browse-collections/

ELECTRONIC HEALTH RECORDS (EHRs): Electronic health records databases incluse both structured and unstructured data and can vary by source. Some form of normalization is probably required.
EXAMPLES:
https://starr.stanford.edu/data-types/electronic-health-record
https://www.nature.com/articles/s41597-022-01899-x

LAB RESULTS DATABSES: This type of data consists of lab results from blood tests, stool tests, urine tests, etc. From my experience these are the hardest to get reliable data from as labs are often private organizations.
https://pmc.ncbi.nlm.nih.gov/articles/PMC2929542/

SKLEARN: The algorithms in this paper will utilize SKLearn's openly available datasets for training models. This is simply the easiest and most compatible data for the small scale examples in this paper.
https://scikit-learn.org/stable/datasets.html#datasets

Feature reduction:

This algorithm utilizes a divide and conquer approach to reduce the number of features while maintain accuracy. This allows for the model to run more efferently as it no longer has to process a high dimensionality of data. Here is the approach:
The time complexity of the feature reduction can be considered m (log reg model epochs) * n (size of data) * log(n) (reduction) or **O(m*n*log(n))**

```
def divide_and_conquer_feature_selection(X_train, y_train, X_test, y_test, features,
target_num_features):
    if len(features) <= target_num_features:
        return features
```

```
    # Divide the features into two
    mid = len(features) // 2
    left = features[:mid]
    right = features[mid:]

    # Evaluate
    aleft = helper(X_train, y_train, X_test, y_test, left)
    aright = helper(X_train, y_train, X_test, y_test, right)

    # Keep the better half
    selected = left if aleft > aright else right

    # Recurse
    return divide_and_conquer_feature_selection(X_train, y_train, X_test, y_test,
selected, target_num_features)

def helper(X_train, y_train, X_test, y_test, feature_subset):
    model = LogisticRegression(max_iter=10000, solver='liblinear')
    model.fit(X_train[feature_subset], y_train)
    y_pred = model.predict(X_test[feature_subset])
    return accuracy_score(y_test, y_pred)
```

END OF ALGORITHM

III. APPROACH

There are 4 general approaches that will be covered in this paper. These include Supervised Learning, Unsupervised Learning, Deep Learning, Statistical Modeling, and Custom Algorithms.

1. SUPERVISED LEARNING:

Supervised learning algorithms utilize datasets with the correct output each data point labeled. By utilizing labeled data, the model can be trained Via backpropagation and can be used on new unseen patient data. Some applications of this type of model include image analysis, radiology, analyzing lab results, and suggesting treatments to certain ailments. Some examples of supervised learning include logistical regression state vector machines and neural networks. By setting the output classes of a neural network to all the possible diseases being tested for, a sufficiently deep neural network utilizing logistical regression may be able to predict diagnosis once trained.

2. UNSUPERVISED LEARNING:

Unsupervised learning is similar to supervised learning with the difference being that data points are not labeled. In this way the model must group data and generate classes based on similarity this type of model can be used to detect anomalies and also classify results with predefined groups. An example of this type of algorithm is a K-Means algorithm. With sufficient generated groups, the model could predict the output of a given datapoint by finding its closest group. It could also detect new groups based on previously unseen trends period

3. STATISTICAL MODELING:

Statistical modeling is a more classical approach to medical diagnosis. These types of models take into account the patients age, symptoms, and medical history to make predictions of what the diagnosis

may be via probability. By asking the patient a series of questions, and having storing the conditional probability of each, the model can predict with some accuracy the general diagnosis of a patients illness.

4. CUSTOM ALGORITHMS:

In most cases, when trying to create a system that can reliably predict the diagnosis, a combination of the previous 3 approaches is used. By integrating the three approaches, we can reliably predict the diagnosis of a patient to aid in medical treatment. The tradeoff of this approach is it is the most computationally / time demanding as it would require the use of 3 or more models. These algorithms will not be covered in this paper as they encompass a wide range of solutions.

IV. SUPERVISED LEARNING ALGORITHM

This section contains an example of a basic supervised learning algorithm that could be used to predict if a sample is malignant. The dataset used contains 569 samples classed as either Malignant or Benign, with 212 samples the former and 357 the latter. This algorithm trains a neural network classifier on the data logs the accuracy of the algorithm.

Complexity:

Determining the complexity of a supervised learning algorithm is not very straight forward depending on the type of algorithm. For this implementation, we can consider training and testing as 2 separate events and thus their complexities are distinct.

Training: Training the algorithm consists of passing the dataset through the network and back-progating to update weights. (2n). We do this on all features (k) and repeat this as many times as we need to reach our target epochs (m). Thus, the time complexity of **training** this supervised algorithm is O(m(n*k)) where n = len(dataset), k = len(features), and m = len(epochs).

Prediction: Predicting if a sample is malignant or not does not require any use of the dataset n. The algorithm utilizes the learned weights to pass the data once through the network and arrive at a result.

Thus the time complexity of **prediction** is linear, at O(1).

Dataset:
https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html
Guides used:
https://www.youtube.com/watch?v=z1oDlzngvI0
https://www.youtube.com/watch?v=EAGeDyygilM

```
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report
import numpy as np


def load_and_prepare_data():
    # Load the dataset
    cancer = load_breast_cancer()
```

```
    X = cancer.data
    y = cancer.target

    # Split the data into train, test, adn validation sets
    # lines 19 and 20 borrowed from sklearn website
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
    X_train, X_val, y_train, y_val = train_test_split(X_train, y_train,
test_size=0.25, random_state=42) # 0.25 of 0.8 is 0.2

    # normalize the data
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_val = scaler.transform(X_val)
    X_test = scaler.transform(X_test)

    # Convert data to tensors
    X_train = torch.tensor(X_train, dtype=torch.float32)
    X_val = torch.tensor(X_val, dtype=torch.float32)
    X_test = torch.tensor(X_test, dtype=torch.float32)
    y_train = torch.tensor(y_train, dtype=torch.long)
    y_val = torch.tensor(y_val, dtype=torch.long)
    y_test = torch.tensor(y_test, dtype=torch.long)

    return X_train, X_val, X_test, y_train, y_val, y_test, scaler


# 2. Define the Model
class BinaryClassifier(nn.Module):

    def __init__(self, input_size):

        super(BinaryClassifier, self).__init__()
        self.fc1 = nn.Linear(input_size, 64)  # layer 1
        self.relu1 = nn.ReLU()
        self.fc2 = nn.Linear(64, 32)       # layer 2
        self.relu2 = nn.ReLU()
        self.fc3 = nn.Linear(32, 1)       #  layer 3 (output layer)
        self.sigmoid = nn.Sigmoid()       # map to value between 0 and 1

    def forward(self, x):
        # forward propigate through network
        out = self.fc1(x)
        out = self.relu1(out)
        out = self.fc2(out)
        out = self.relu2(out)
        out = self.fc3(out)
        out = self.sigmoid(out)
```

```
        return out



def train_model(model, X_train, y_train, X_val, y_val, epochs=100,
learning_rate=0.001):

    criterion = nn.BCELoss()
    optimizer = optim.Adam(model.parameters(), lr=learning_rate)

    # Train the model
    val_accuracies = []
    for epoch in range(epochs):
        # Forward propagation
        y_pred = model(X_train)
        loss = criterion(y_pred, y_train.float().view(-1, 1))

        # Backward propagation and optimization
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        # Evaluate on validation set
        with torch.no_grad():
            y_pred_val = model(X_val)
            y_pred_val_binary = (y_pred_val > 0.5).long()  # Threshold at 0.5
            val_accuracy = accuracy_score(y_val.numpy(), y_pred_val_binary.numpy())
            val_accuracies.append(val_accuracy)

        # Print training progress
        print(f'Epoch {epoch+1}/{epochs}, Loss: {loss.item():.4f}, Validation
Accuracy: {val_accuracy:.4f}')
    return val_accuracies

def predict(model, scaler, data):
    # Convert the input data to a numpy array if it's a list
    if isinstance(data, list):
        data = np.array(data).reshape(1, -1)  # Reshape to (1, num_features)
    elif isinstance(data, np.ndarray):
        if data.ndim == 1:
          data = data.reshape(1, -1)
    else:
        raise TypeError("Input data must be a list or a numpy array.")

    # Scale the data using the scaler fitted on the training data
    data_scaled = scaler.transform(data)
    # Convert the scaled data to a tensor
    data_tensor = torch.tensor(data_scaled, dtype=torch.float32)
```

```
    # Make the prediction
    model.eval()  # Set the model to evaluation mode
    with torch.no_grad():  # Disable gradient calculation for inference
        y_pred_prob = model(data_tensor)
        y_pred_binary = (y_pred_prob > 0.5).long()  # Threshold at 0.5

    return y_pred_binary.item()  # Return the prediction as a Python integer

def evaluate_model(model, X_test, y_test):
    # Evaluate the model on the test set
    with torch.no_grad():
        y_pred_test = model(X_test)
        y_pred_test_binary = (y_pred_test > 0.5).long()  # Threshold at 0.5
        test_accuracy = accuracy_score(y_test.numpy(), y_pred_test_binary.numpy())

    print(f'Test Accuracy: {test_accuracy:.4f}')
    print(classification_report(y_test.numpy(), y_pred_test_binary.numpy()))

if __name__ == "__main__":
    # Load and prepare the data
    X_train, X_val, X_test, y_train, y_val, y_test, scaler = load_and_prepare_data()

    # Define the model
    input_size = X_train.shape[1]  # Number of features
    model = BinaryClassifier(input_size)

    # Train the model
    val_accuracies = train_model(model, X_train, y_train, X_val, y_val, epochs=100,
learning_rate=0.001)

    # Evaluate the model
    evaluate_model(model, X_test, y_test)
```

END OF ALGORITHM

RESULTS:

Metric	Benign	Malignant	Accuracy	Macro Av	Weighted Avg
Precision	0.98	0.99	0.98	0.98	0.98

Recall	0.98	0.99			
F1-Score	0.98	0.99			
Count	43	71			

Figure 1. Supervised Learning Results

CONFUSION MATRIX:

Supervised Learning Results			
TARGET / OUTPUT	Benign	Malignant	SUM
Benign	42 36.84%	1 0.88%	43 97.67% 2.33%
Malignant	3 2.63%	68 59.65%	71 95.77% 4.23%
SUM	45 93.33% 6.67%	69 98.55% 1.45%	110 / 114 96.49% 3.51%

Figure 2. Confusion matrices generated via https://www.damianoperri.it/public/confusionMatrix/

V. UNSUPERVISED LEARNING ALGORITHM

This section contains an example of a basic unsupervised learning algorithm that could be used to predict if a sample is malignant. This algorithm utilizes the same exact dataset as before, but the algorithm does not utilize the labels for training. This algorithm (K-MEANS) creates cluster centers, assigns data based on how close the datapoint is to a given cluster, then re-averages the cluster centers and re-assigns the data again. This way the clusters become more and more fitting of the data it represents. The clusters are then matched with their actual values.

Complexity:

As with Supervised, Unsupervised learning is also broken down into training and testing complexities.

Training: For training, we are passing each datapoint (n) into our graph checking each cluster (2) based on the number of features (m), and we are doing this as many times as we need until our clusters stop changing (for sklearn this runs 10 times, but for demonstration purposeless we will call this k). Thus our **training** complexity is n * 2 * m * k, or simply O(n*m*k)

Prediction: Predicting our datapoint in this unsupervised algorithm is not linear as it was in our supervised algorithm. It checks each cluster (2) by the number of features in the sample (n), this for our **prediction** approach our complexity is 2n or O(n), but in cases where the number of clusters is not simple, we can cay O(n*m)

Dataset:
https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html

```
import numpy as np
from sklearn.datasets import load_breast_cancer
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# load the data the same way as the supervised data but without labels
def load_and_prepare_data():
    # Load the dataset
    cancer = load_breast_cancer()
    X = cancer.data
    y = cancer.target #useful for comparing the clusters to the actual labels

    # note that this alg only loads the X values, not the Y values (no labels)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    return X_train, X_test, y_train, y_test, scaler
```

```
# train a k-means model (creates clusters and matches each datapoint to the cluster it
is closest to)
def train_kmeans_model(X_train, n_clusters=2):

    kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init='auto')
    # Train the model
    kmeans.fit(X_train)
    return kmeans

def evaluate_kmeans_model(kmeans, X_test, y_test):
    # Predict the clusters for the test data
    cluster_labels = kmeans.predict(X_test)
    assigned_clusers = assign_clusters_to_labels(kmeans, X_test, y_test)
    if assigned_clusers[0] == 1: # if the labels are flipped, flip the labels of each
        for i in range(len(cluster_labels)):
            if cluster_labels[i] == 1:
                cluster_labels[i] = 0
            else :
                cluster_labels[i] = 1
    # generate variables for confusion matrix (same as before)
    tp =0
    fp= 0
    fn=0
    tn=0
    for i in range(len(cluster_labels)):
        if cluster_labels[i] == 1:
            if y_test[i] == 1:
                tp+=1
            else:
                fp+=1
        else:
            if y_test[i] == 1:
                fn+=1
            else:
                tn+=1
    print("fn", fn ," fp ", fp , " tn " , tn , " tp ", tp)
    #print(cluster_labels, y_test, assigned_clusers)

# goes through each cluster and finds the majority value (0 or 1) then assigns that
cluster that valeu
def assign_clusters_to_labels(kmeans, X_train, y_train):

    clusters = kmeans.predict(X_train)
    cluster_labels = {}
    for cluster_id in range(kmeans.n_clusters):
        cluster_indices = np.where(clusters == cluster_id)[0]
        cluster_true_labels = y_train[cluster_indices]
```

```
        # Find the most frequent label in this cluster
        if cluster_true_labels.size > 0:
            most_common_label = np.bincount(cluster_true_labels).argmax()
        else:
            most_common_label = 0
        cluster_labels[cluster_id] = most_common_label
    return cluster_labels

def predict_with_kmeans(kmeans, cluster_labels_map,  data):

    # Convert the input data to a numpy array if it's a list (same as before)
    if isinstance(data, list):
        data = np.array(data).reshape(1, -1)  # Reshape to (1, num_features)
    elif isinstance(data, np.ndarray):
        if data.ndim == 1:
          data = data.reshape(1, -1)
    else:
        raise TypeError("Input data must be a list or a numpy array.")

    # Predict the cluster for the new data point
    cluster = kmeans.predict(data)[0]
    # Get the label for that cluster
    predicted_label = cluster_labels_map[cluster]
    return predicted_label

if __name__ == "__main__":
    # Load and prepare the data
    X_train, X_test, y_train, y_test, scaler = load_and_prepare_data()

    # Train the K-Means model
    n_clusters = 2  # benign or malig
    kmeans_model = train_kmeans_model(X_train, n_clusters=n_clusters)

    # Evaluate the model
    evaluate_kmeans_model(kmeans_model, X_test, y_test)

    # Assign labels to clusters
    cluster_labels_map = assign_clusters_to_labels(kmeans_model, X_train, y_train)
    print(f"Cluster to label mapping: {cluster_labels_map}")
```

END OF ALGORITHM

RESULTS:

Metric	Benign	Malignant	Accuracy	Macro Av	Weighted Avg
Precision	0.97	0.909	0.9298	0.9230	0.9283
Recall	0.837	0.985			
F1-Score	0.90	0.945			
Count	43	71			

Figure 3. Unsupervised Learning Results

CONFUSION MATRIX:

UnSupervised Learning Results			
TARGET / OUTPUT	Benign	Malignant	SUM
Benign	36 31.58%	1 0.88%	37 97.30% 2.70%
Malignant	7 6.14%	70 61.40%	77 90.91% 9.09%
SUM	43 83.72% 16.28%	71 98.59% 1.41%	106 / 114 92.98% 7.02%

Figure 4. Confusion matrices generated via https://www.damianoperri.it/public/confusionMatrix/

VI. STATISTICAL MODELING & CUSTOM

Statistical Modeling is another method of utilizing models to predict a diagnosis. An excellent example can be found in the paper, *Evolving a Bayesian classifier for ECG-based age classification in medical applications by M. Wiggins , A. Saad, B. Litt , G. Vachtsevanos (Reference 22).*

In this paper, the researchers utilized a Bayes Classifier for age based classification. Bayes Classifiers work by utilizing prior probabilities to calculate the probability that a data sample belongs in any given class. It does this by multiplying the probability that the data point is in the group (by using the groups frequency over N) then multiplies this by the likelihood of each feature of the example appearing in that group. When done for every group, we get a set of probabilities, and we choose the highest value as our prediction.

Complexity:

As before, we will split the complexity into training complexity and predicting complexity.

Training: For training, we utilize the number of datapoints (n) and for each we utilize each feature (m) and compute the statistics based on these. This we can claim that for our binary approach, the complexity to **train** the model would be O(n*m). This is significantly faster training than our other two models.

Prediction: Predicting our datapoint in this unsupervised algorithm is not linear as it was in our supervised algorithm. It checks each cluster (2) by the number of features in the sample (n), this for our **prediction** approach our complexity is 2n or O(n), but in cases where the number of clusters is not simple, we can cay O(n*m)

RESULTS:

Method	**Accuracy**
AUC	0.8475
GA	0.8175
K2	0.65
AVERAGE	0.771

Figure 5. Statistical Modeling Results

We can clearly see that of the three models, Statistical performed the worst. This can be attributed to the fact that this type of model does not ‘learn’ trends like a supervised model but rather utilizes the probabilities of those trends.

Custom algorithms can be utilized to cross reference the results of each model. For example, a custom algorithm could run several models and use a voting system to ensure the highest likelihood of having an accurate result. In this approach, we could utilize the previously trained Supervised and Unsupervised algorithms and put all three predictions as votes. If there is a disagreement (2 positives and 1 negative, 2 negatives and 1 positive, etc), the higher frequency result would be the overall output.

VII. Conclusion

When surveying the algorithms for patient diagnosis, we can conclude that Supervised Learning, Unsupervised Learning, and Statistical Modeling algorithms can accurately and consistently detect and diagnose diseases in patients. The implications of these algorithms can be seen in online AI based wellness applications, as well as in the medical field. This technology has already begun testing in radiology and cancer detection applications in the modern medical field. While some algorithms are more accurate than others, there is an increased benefit when using multiple algorithms in collaboration to increase overall accuracy. This technology will likely change the medical landscape and allow for faster, cheaper, and more accurate diagnosis, ultimately making healthcare more accessible and affordable.

VIII. References

SUPERVISED LEARNING

1. Buch, V. H., Ahmed, I., & Maruthappu, M. (2018). Artificial intelligence in medicine: current trends and future possibilities. *The British journal of general practice : the journal of the Royal College of General Practitioners*, *68*(668), 143–144. https://doi.org/10.3399/bjgp18X695213

2. Lei Wang, Qing Qian, Qiang Zhang, Jishuai Wang, Wenbo Cheng, Wei Yan, Classification Model on Big Data in Medical Diagnosis Based on Semi-Supervised Learning, *The Computer Journal*, Volume 65, Issue 2, February 2022, Pages 177–191, https://doi.org/10.1093/comjnl/bxaa006

3 .A. Chebli, A. Djebbar and H. F. Marouani, "Semi-Supervised Learning for Medical Application: A Survey," *2018 International Conference on Applied Smart Systems (ICASS)*, Medea, Algeria, 2018, pp. 1-9, doi: 10.1109/ICASS.2018.8651980. https://www.sciencedirect.com/science/article/abs/pii/S0933365715000482

4 . Ramakanth Kavuluru, Anthony Rios, Yuan Lu,
An empirical evaluation of supervised learning approaches in assigning diagnosis codes to electronic medical records, https://www.sciencedirect.com/science/article/abs/pii/S0933365715000482

5. Roy, S., Meena, T., & Lim, S.-J. (2022). Demystifying Supervised Learning in Healthcare 4.0: A New Reality of Transforming Diagnostic Medicine. *Diagnostics*, *12*(10), 2549. https://doi.org/10.3390/diagnostics12102549

6. Samriti Sharma, Gurvinder Singh, Manik Sharma, A comprehensive review and analysis of supervised-learning and soft computing techniques for stress diagnosis in humans.
https://www.sciencedirect.com/science/article/abs/pii/S0010482521002444

7. Eckardt Jan-Niklas , Bornhäuser Martin , Wendt Karsten , Middeke Jan Moritz, Semi-supervised learning in cancer diagnostics, Volume 12 – 2022,
https://www.frontiersin.org/journals/oncology/articles/10.3389/fonc.2022.960984 DOI=10.3389/fonc.2022.960984

8. Aljuaid, A., Anwar, M. Survey of Supervised Learning for Medical Image Processing. *SN COMPUT. SCI.* **3**, 292 (2022).

9. S. Muthurajkumar, R. Praveen, A.A. Abd El-Aziz, R. Shangeeth, S. Anika Lakshmi, and R. Gaythrisr, Medical diagnosis of human heart diseases using supervised learning techniques, https://digital-library.theiet.org/doi/abs/10.1049/PBHE058E_ch4

10. M. Zhao, R. H. M. Chan, T. W. S. Chow and P. Tang, "Compact Graph based Semi-Supervised Learning for Medical Diagnosis in Alzheimer's Disease," in *IEEE Signal Processing Letters*, vol. 21, no. 10, pp. 1192-1196, Oct. 2014

11. Oumaima Terrada1 , Soufiane Hamida1 , Bouchaib Cherradi1, 2, Abdelhadi Raihani1,* , Omar Bouattane1, Supervised Machine Learning Based Medical Diagnosis Support System for Prediction of Patients with Heart Disease, https://www.researchgate.net/profile/Omar-Bouattane/publication/345325800_Supervised_Machine_Learning_Based_Medical_Diagnosis_Support_System_for_Prediction_of_Patients_with_Heart_Disease/links/5fb660c9458515b797511e8d/Supervised-Machine-Learning-Based-Medical-Diagnosis-Support-System-for-Prediction-of-Patients-with-Heart-Disease.pdf

12. Nosayba Al-Azzam, Ibrahem Shatnawi, Comparing supervised and semi-supervised Machine Learning Models on Diagnosing Breast Cancer, Annals of Medicine and Surgery, https://academic.oup.com/comjnl/article-abstract/65/2/177/5808795

13. M. Pechenizkiy, A. Tsymbal, S. Puuronen and O. Pechenizkiy, "Class Noise and Supervised Learning in Medical Domains: The Effect of Feature Extraction," 19th IEEE Symposium on Computer-Based Medical Systems (CBMS'06), Salt Lake City, UT, USA, 2006, pp. 708-713, doi: 10.1109/CBMS.2006.65. https://ieeexplore.ieee.org/abstract/document/1647654

14. De Asmundis, R., Guarracino, M.R. (2013). Mathematical Models of Supervised Learning and Application to Medical Diagnosis. In: Pardalos, P., Coleman, T., Xanthopoulos, P. (eds) Optimization and Data Analysis in Biomedical Informatics. Fields Institute Communications, vol 63. Springer, New York, NY. https://doi.org/10.1007/978-1-4614-4133-5_3

15. Haseeb Hassan, Zhaoyu Ren, Chengmin Zhou, Muazzam A. Khan, Yi Pan, Jian Zhao, Bingding Huang, Supervised and weakly supervised deep learning models for COVID-19 CT diagnosis: A systematic review, https://www.sciencedirect.com/science/article/pii/S0169260722001171

UNSUPERVISED LEARNING:

16. Khalid Raza and Nripendra K. Singh, A Tour of Unsupervised Deep Learning for Medical Image Analysis, https://www.benthamdirect.com/content/journals/cmir/10.2174/1573405617666210127154257

17. Xiuli Bi, Shutong Li, Bin Xiao, Yu Li, Guoyin Wang, Xu Ma,
Computer aided Alzheimer's disease diagnosis by an unsupervised deep learning technology. https://www.sciencedirect.com/science/article/abs/pii/S0925231219304709

18. H. P. Ng, S. H. Ong, K. W. C. Foong, P. S. Goh and W. L. Nowinski, "Medical Image Segmentation Using K-Means Clustering and Improved Watershed Algorithm," *2006 IEEE Southwest Symposium on Image Analysis and Interpretation*, Denver, CO, USA, 2006, pp. 61-65, doi: 10.1109/SSIAI.2006.1633722. https://ieeexplore.ieee.org/abstract/document/1633722

STATISTICAL MODELS:

19. Langarizadeh, M., & Moghbeli, F. (2016). Applying Naive Bayesian Networks to Disease Prediction: a Systematic Review. *Acta informatica medica : AIM : journal of the Society for Medical Informatics of Bosnia & Herzegovina : casopis Drustva za medicinsku informatiku BiH*, *24*(5), 364–369. https://doi.org/10.5455/aim.2016.24.364-369

20. Regnier-Coudert O, McCall J, Lothian R, Lam T, McClinton S, N'dow J. Machine learning for improved pathological staging of prostate cancer: a performance comparison on a range of classifiers. Artif Intell Med. 2012 May;55(1):25–35. doi: 10.1016 https://www.sciencedirect.com/science/article/abs/pii/S0933365711001461

21 .Kazmierska J, Malicki J. Application of the Naïve Bayesian Classifier to optimize treatment decisions. Radiother Oncol. 2008 Feb;86(2):211–6. doi: 10.1016/j.radonc.2007.10.019 https://www.sciencedirect.com/science/article/abs/pii/S0167814007005221

22. M. Wiggins , A. Saad , B. Litt , G. Vachtsevanos, Evolving a Bayesian classifier for ECG-based age classification in medical applications, https://www.sciencedirect.com/science/article/abs/pii/S1568494607000452

23. Beí¡ta Reiz, Lehel Csató, Bayesian Network Classifier for Medical Data Analysis, https://www.univagora.ro/jour/index.php/ijccc/article/view/2414

24. KONONENKO, I. (1993). INDUCTIVE AND BAYESIAN LEARNING IN MEDICAL DIAGNOSIS. Applied Artificial Intelligence, 7(4), 317–337. https://doi.org/10.1080/08839519308949993

25. A. F. M. Hani, H. A. Nugroho and H. Nugroho, "Gaussian Bayes classifier for medical diagnosis and grading: Application to diabetic retinopathy," 2010 IEEE EMBS Conference on Biomedical Engineering and Sciences

(IECBES), Kuala Lumpur, Malaysia, 2010, pp. 52-56, doi: 10.1109/IECBES.2010.5742198.
https://ieeexplore.ieee.org/abstract/document/5742198