Chalmers Tekniska Högskola

CHALMERS
UNIVERSITY OF TECHNOLOGY

DESIGN OF AI SYSTEMS (DAT410)

Module 5: Diagnostic Systems

Group 38

Himanshu Sahni
9812075498
sahni@chalmers.se
MPCAS

Nina Uljanić
9609134920
uljanic@chalmers.se
MPCAS

We hereby declare that we have both actively participated in solving every exercise. All solutions are entirely our own work, without having taken part in other solutions.

Hours:
Himanshu Sahni: 20h
Nina Uljanić: 15h

1. Reading and reflection

Summary of "The Mythos of Model Interpretability"

The author begins by stressing the significance of interpretability and provides a list of concepts that are outlined in this article. The author asserts that models must not only perform well but must also be interpretable. However, the definition of interpretability is not straightforward, as numerous authors have suggested different interpretations. The primary motive behind the need to establish a widely accepted interpretation of this term is to comprehend how models operate and to provide a suitable response to the question of "why did the model assign that value to that input?"

The author has discussed various different properties related to interpretability:

- **Transparency** - It is defined as the ease to understand what a model does.
- **Trust** - It can be defined as the ease to understand a model or the capability of a system to generate an accurate but unbiased output.
- **Causality** - It can be defined as an approach to defining a hypothesis that can be tested. Usually, the practitioners do not rely on casual reference in case of large and complex data sets.
- **Transferability** - It is defined as the ability of a model to be deployed in different situations, also known as the ability to generalize.
- **Informativeness** - This refers that in the real world, the purpose of an ML system is to provide useful information which is not limited to the output.
- **Fair and ethical decision making** - The objective of the ML systems from this perspective is to generate output without gender, race, or any group bias.

In the paper, the author has not been able to fix a particular definition for interpretability but he has proposed a set of properties to confer the interpretability to a model and has classified them into two categories:

- **Transparency related properties:**
  - Simulability - It can be defined as the trade-off between the complexity in terms of computational resources and the size of the model.
  - Decomposability - It can be defined as the possibility of each part(input parameters and calculation) as an intuitive explanation.
  - Algorithmic transparency: - It can be defined as the ease of an algorithm interface i.e., The ability to provide an intuitive explanation of the operations carried out by an algorithm.
- **Post hoc interpretability:** It refers to the approach used to confer useable information for users and practitioners.
  - Text explanations - It refers to the generation of text intended to support the main output.
  - Visualization - It refers to rendering the visualizations with the aim to learn what a model has learned.
  - Local explanations - It refers to explaining the decisions in detail made by a model at every step.

– Explanations by example - It refers to using models with calculations that can be used to make analogies and help in understanding the actions.

The conclusion that can be drawn from this article is that interpretability is not defined properly. It is important to have a clear and balanced evaluation of interpretability between linear and deep learning models and the trade-off between transparency and decomposability should be accounted for.

Summary of "Machine learning techniques to diagnose breast cancer from image-processed nuclear features of the fine needle aspirates (FNA)"

The research aimed to investigate the use of machine learning techniques to diagnose breast cancer by analyzing the image-processed nuclear features of fine needle aspirates (FNAs), a minimally invasive biopsy technique. The paper reports that the most accurate results were obtained by using a ten-fold cross-validation method with 569 samples as the training set and 54 samples as the test set. The accuracy of the results and the preparation of samples for FNA are also discussed. To ensure accurate digital assessment, the images were prepared to minimize nuclear overlap, and pinpointing the precise location of the cell nucleus was a crucial step in the process.

The author has explained the method employed to convert the images into usable data. Specifically, the method utilized is known as "snakes," which is an active contour model designed to minimize an energy function based on the arc length of a curve, resulting in a splined output. Furthermore, the author defines and elaborates on nuclear characteristics derived from this model. These 10 characteristics are described below:

- Radius - The average length between each point on the snake and the center of the cell is used to determine the radius.
- Perimeter - The nuclear perimeter is defined as the sum of the distances between each consecutive point on the snake.
- Area - The number of pixels within the snake's interior is counted, and then half the number of pixels on the perimeter are added to the count.
- Compactness - Measured by dividing the square of the perimeter by the area. It increases with the irregularity of the boundary.
- Smoothness - The difference between the length of a radial line and the average length of the lines that surround it.
- Concavity - In this, the length to the center of the cell is measured by drawing some random low-length chords between the point of the snake. The higher the length, the higher the concavity.
- Concave points - Same as concavity but only counts the number.
- Symmetry - The major axis of the nucleus i.e. the longest chord passing through its center is determined initially and then the difference in length between lines perpendicular to the major axis and the nuclear boundary in both directions is then measured. If the major axis cuts the boundary due to a concavity, special attention is paid to account for this.
- Fractal dimension - Here different scales are used to measure the perimeter and then a plot is made between the log of measured values and the log of scales. The slope of this plot is the fractal dimension.

- Texture - It is the variance of the grayscale intensities in the component pixels is calculated.

We can see that there were two main features that were measured i.e., size and shape. The classification method used was MSM-tree which uses linear programming iteratively similar to SVM. The methodology used to achieve generalization was to minimize the planes and the number of features.

2. Implementation

In this task, we implemented three classifiers to assign the diagnosis of malignant or benign to a dataset of patients tested for breast cancer, with each patient represented by 10 different features (training) and a diagnosis (target). The implementation consists of three steps: 1) a rule-based classifier, 2) a random forest classifier, and 3) a decision tree classifier. Also, we used K-fold cross-validation so that our model fits the data well, and we can build a better classifier. The reason for using the StratifiedKFold is that it is better than the Kfold because StratifiedK-Fold takes into account the class distribution and ensures that each fold has approximately the same proportion of each class as the whole dataset. On the other hand, KFold simply divides the data into k equal-sized folds and uses each fold as a validation set, which can result in a bias towards the majority class if there is a significant class imbalance.

Rule based classifier

To accomplish the given task, a large number of dimensions were provided, and it's crucial to define the variables that will be used to establish the rules accurately. This is necessary to ensure that the required points are correctly classified. Although there are various methods to do this, filtering the given dimensions effectively appears to be the optimal approach, according to our opinion. Since we are not an expert in this topic we used the available information appropriately by reviewing relevant literature. After going through the recommended article, we found that the authors explicitly and implicitly defined which features were intended to describe or measure a property in their discussion. For example, radius and area intend to measure the size, the texture is used for texture, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension intend to measure the shape.

As mentioned, it was required to build a rule-based classifier with a set of certain rules stating: If (cell size is abnormal]) Or (cell shape is abnormal) Or (cell texture is abnormal) Or (cell similarity/homogeneity is abnormal) then: diagnosis is malignant otherwise: diagnosis is benign.

For our implementation we choose the following features:

1. for cell size - perimeter_0
2. for cell shape - compactness_2
3. for cell homogeneity - concave points_2
4. for cell texture - texture_2

Here we tried to analyze the tradeoff between specificity and sensitivity, using a graphical density plot for each of the selected features: Our idea here is to use the features as rules to separate the dataset into benign and malignant points. This is one based on the shared area between the graphs. For example, for cell size' and for 'cell similarity we have used the perimeter and the concave points_2 features as rules as indicated by the overlapping areas of the density plots for each group. This suggests that these selected features are effective in differentiating between the two classes. Similarly, we tried the combination of different properties and visualized these graphs, and choose the above-mentioned 4 features for our classifier.

It's crucial to note that the previous choice was made because the decision criteria are interconnected by an "or" condition. This implies that if any one of them fails during the classification process, our classifier will produce a false positive. In making this selection, we relied on the density plots and the "or" clause limitation provided in the exercise for our criteria.

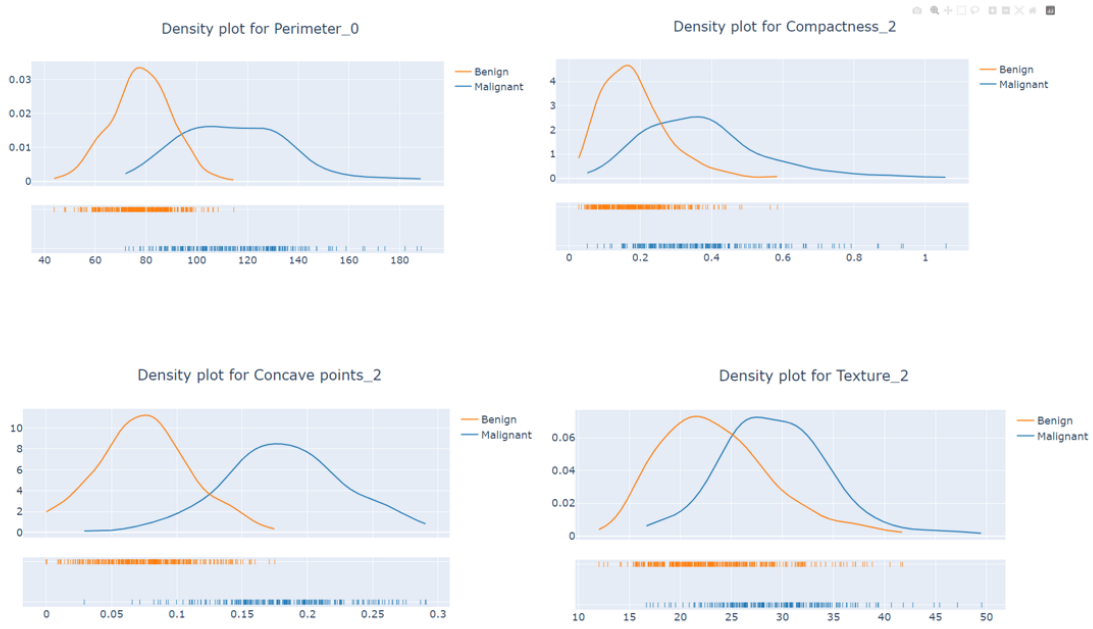

Figure 1: density plots for the selected features.

The "or" clause restriction is very important because if the values are set incorrectly, it could lead to an excessive number of false positives being classified, as we previously discussed.
We have used certain values (thresholds) for our classifier, these values have been found from the graph. The objective is to choose a value that maximizes specificity without significantly reducing sensitivity.

Rule-based classifier

For the rule-based classifier, we selected the four aforementioned features from the data set for consideration. The mean accuracy of our rules-based implementation is 88%. Table 1 shows the accuracy measured for each of the four iterations of the stratified k-fold cross-validation. For a graphical representation of the results, see Figure 2.

Accuracy	Mean
0.90	0.88
0.85	
0.91	
0.85	

Table 1: Accuracy values for the rule-based classifier, and the mean.

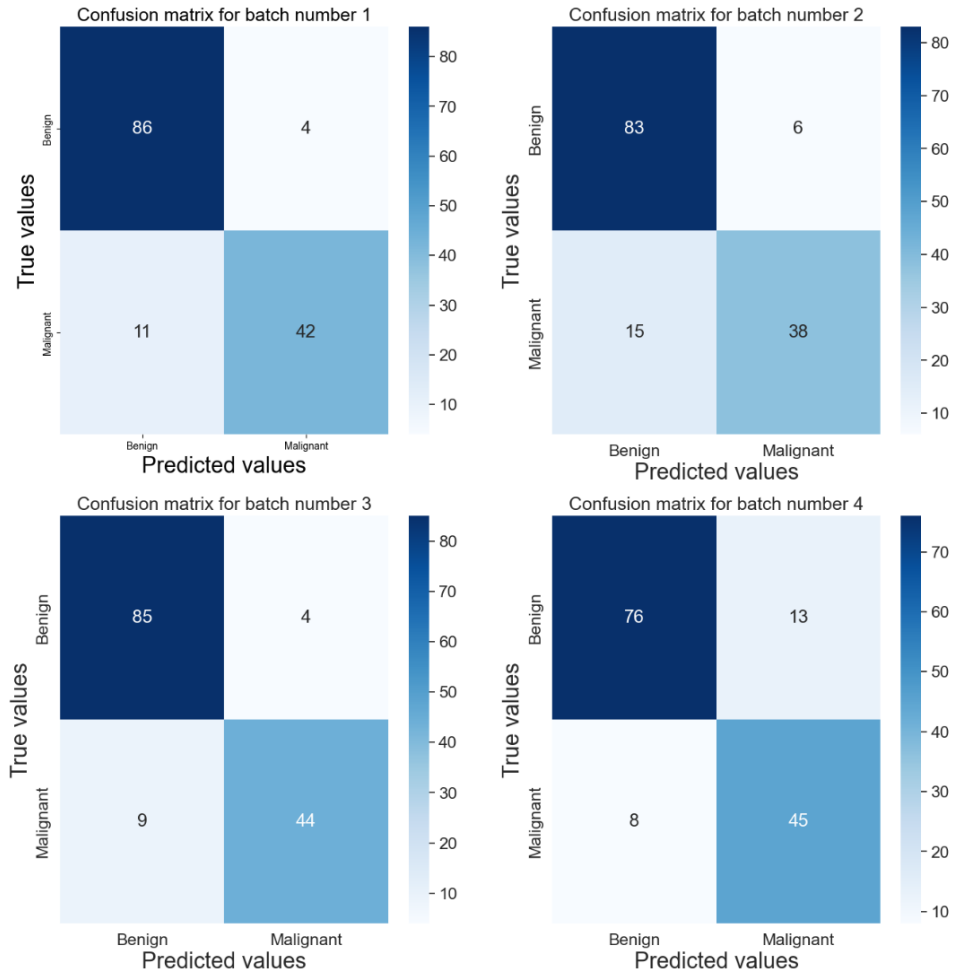

Figure 2: Graphical representation of the results for the rule-based classifier.

Random forest classifier

The implementation of the random forest classifier has a similar structure as that of the rule-based classifier mentioned above, the only difference being that we're calling the sklearn framework function RandomForestClassifier instead of our own rule-based function. The mean accuracy of the random forest classifier is 96%. Table 2 shows the accuracy measured for each of the four iterations. For a graphical representation of the results, see Figure 3.

Accuracy	**Mean**
0.94	0.96
0.96	
0.98	
0.97	

Table 2: Accuracy values for the random forest classifier.

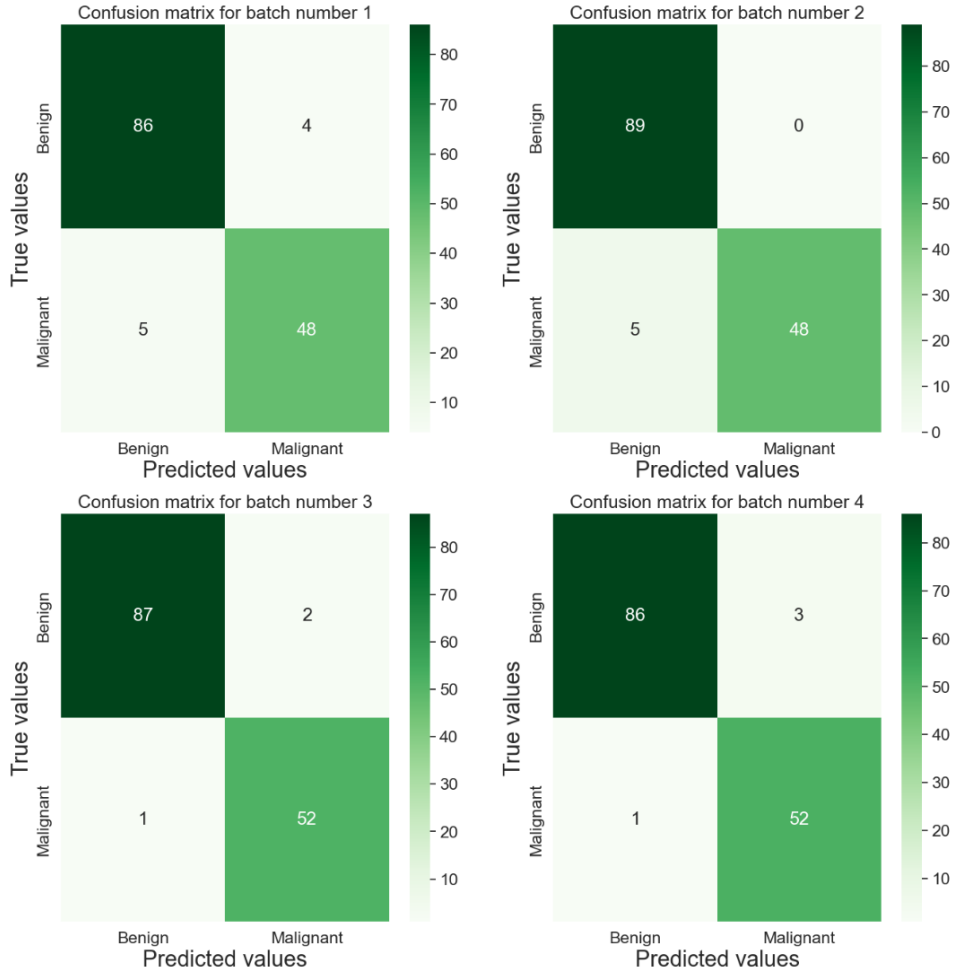

Figure 3: Graphical representation of the results for random forest classifier.

Decision tree classifier

For the third task, where we had to implement a classifier of our own design, we chose to implement a decision tree classifier. The basic idea of a decision tree is to split the data into smaller subsets based on the values of the features, and then recursively apply the same process to each subset until a stopping criterion is met. To this end, we use the sklearn framework DecisionTreeClassifier function. The rest of the code is identical to the second implementation above. The mean accuracy of the random forest classifier is 93%. Table 3 shows the accuracy measured for each of the four iterations. For graphical representation of the results, see Figure 4.

Accuracy	**Mean**
0.91	0.93
0.95	
0.96	
0.90	

Table 3: Accuracy values for the decision tree classifier.

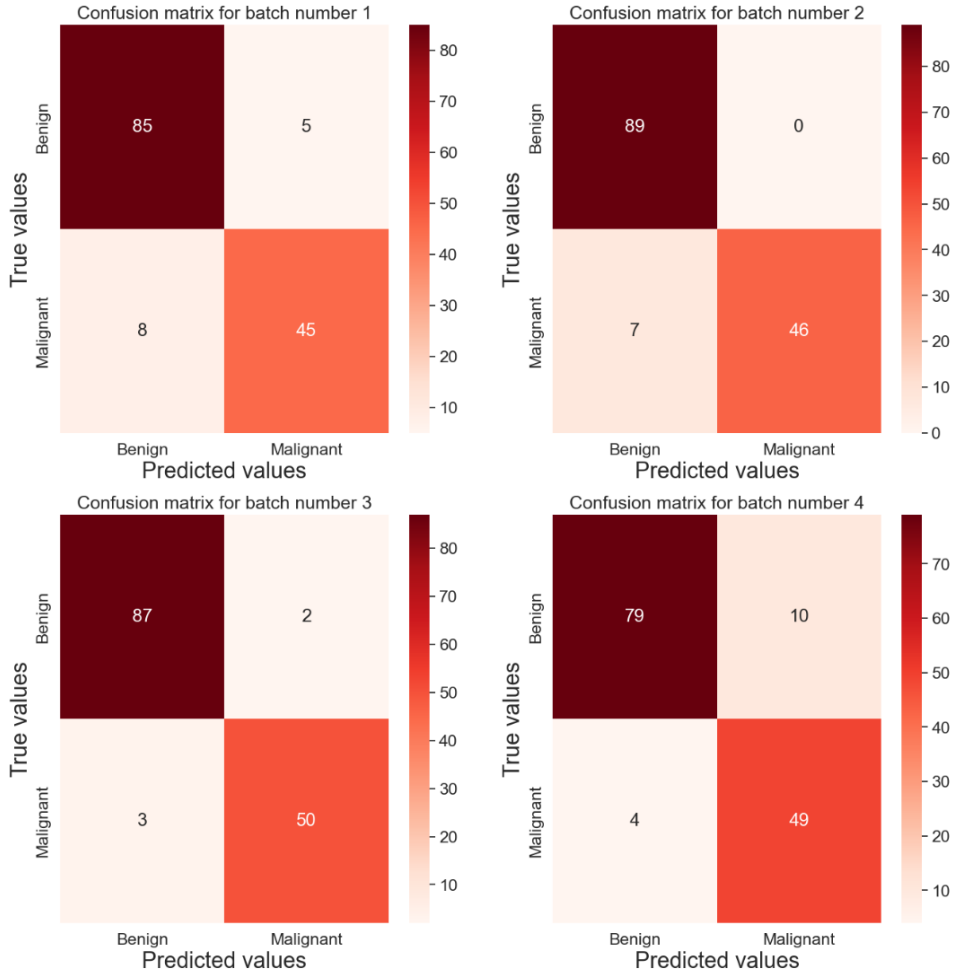

Figure 4: Graphical representation of the results for decision tree classifier.

Discussion

Comparing the three classifiers, we can see that the random forest classifier performs the best, with the decision tree classifier being slightly worse.

Interpretability refers to the ability to understand how an ML model works and makes predictions. It is a particularly important factor in medical diagnosis tools; here it is necessary to explain the rationale behind the model's decisions.

Rule-based classifiers have generally high interpretability because they rely on a set of predefined rules that can easily be understood and explained. They are designed to make decisions based on specific conditions which are usually simple and easy to interpret, making it straightforward to explain how a decision was made. They can be considered as models which can be easily simulated and easily decomposed.

Decision trees are likewise highly interpretable because they are based on a hierarchical structure of decision nodes. One can easily trace the decision-making process of the decision tree, i.e. the steps taken to reach a conclusion. They can also be easily simulated and easily decomposed.

Random forest classifiers are less interpretable than the other two classifiers. They use multiple decision trees to make predictions, and the final prediction is based on the majority vote of individual trees. This makes it difficult to trace the decision-making process of the classifier. In this case, these models are not easily simulated, and not easily decomposed.

Model	**Accuracy ranking**	**Interpretability ranking**
Rule-based	3	1
Random forest	1	3
Decision tree	2	2

Table 4: Accuracy and interpretability ranking of the three classifiers.

3. Discussion

Interpretability of systems, models, and predictions is important in many applications. Healthcare is one of these. Discuss the meaning of interpretability in the context of this week's assignment, its potential benefits and drawbacks, and how interpretability may be defined. Be as specific as you can and use mathematical notation where appropriate. Take inspiration from e.g., "The Mythos of Model Interpretability".

Himanshu Sahni

In my opinion, the purpose of this task was to aid our comprehension of interpretability by means of a basic interactive exercise. The implementation part helped us to understand the meaning of interpretability in an interactive way as we formulated a relatively simple yet effective classifier utilizing rules lined to a "or" clause. In the medical domain, it is important to possess models that are easily understandable, as various pathologies can be linked to an extensive range of illnesses. Interpretability may be deemed as accountability in the context where an inquiry into a specific situation needs to be carried out in order to identify possible areas for improvement or errors.

As we have discussed in the paper "The Mythos of Model Interpretability", defining the concept of interpretability poses a considerable challenge. The method employed by the author of "The Mythos of Model Interpretability" is quite informative and beneficial in presenting a comprehensive perspective, but even he refrains from providing a clear-cut definition of the concept. Instead, the author suggests a series of attributes that contribute to interpretability, categorized into two main groups: transparency and post hoc interpretability.

Nina Uljanić

As explained in the previous section, interpretability is simply the ability to understand how a ML model makes predictions. In healthcare, the stakes are high, and the diagnosis and treatment decisions need to be reliable. Therefore, interpretability is critical. Diagnostic tools can simplify the process of identifying potential health issues and provide more accurate diagnoses. If such a system is interpretable (instead of being black-box), the healthcare workers will be able to understand and trust the decisions made by such systems, greatly increasing their efficiency and decreasing time spent on figuring out causes for patient's symptoms.

The majority of ML systems are black-box. Such systems use well-established algorithms that are simple and cheap to implement. An interpretable model, however, is not. For a system to be interpretable, it oftentimes has to be simplified. However, simiplification might cause it to lose accuracy, which is the parameter that should be maximized. Therefore, a drawback of interpretable models is that they are simple and might not be able to capture all the important nuances in the data.

Individual Summary and Reflection

Lecture 4: Diagnostic Systems (2023-02-14)

Lecture summary by Himanshu Sahni

Diagnosis refers to finding out the cause of a phenomenon and Differential diagnosis is to pick among possible causes. This is performed in different domains like Software debugging, Reasons for user churn, and Failure investigation (anything from cars to infrastructure), however, we focus on the medical domain.
Diagnosis is usually based on the data, it is done based on the symptoms, signs, and tests conducted. A medical diagnosis could be either clinical, laboratory, or radiology. This is used to find the possible causes and these tests are the core of diagnosis. It could be based on passively or actively collected data.
In an ideal scenario, there should be a balance between sensitivity and specificity based on the level of superposition of the density plots and the exposed danger of the subject to the intended diagnosis. Model-based tests are statistical tests that estimate the probability that the assigned label is the right one. These work well in case of a single symptom but usually have poor performance when there are multiple systems. An appropriate example can be the Naives-based model which is based on the assumption that the symptoms are conditionally independent. INTERNIST-1 was one of the first successful commercial diagnostics support systems and it was based on a similar idea as the Naïve Bayes model, but heuristic. It is important to know that the diagnostic test will not be 100 % accurate and therefore we must trade off specificity and sensitivity.
In recent years there have been improvements in the field of machine learning, we have large datasets, several algorithms, and tools to improve the speed and consistency of the system. Data collection has improved and standardization is being performed. The conclusion of that for AI to have an impact on diagnostics it needs to be reproducible, implementable in clinical practice, and efficient.

Module 4 reflection by Himanshu Sahni

In the previous module, we looked into different translation techniques and implemented one of the early models (IBM model 1) for machine translation. The IBM models used word alignments and the expectation-maximization algorithm to determine the most probable sentence. In my view, context and the relation between words are important when translating sentences from one language to another. Also, I think an NLP model is difficult to develop compared to linear models.

Lecture summary by Nina Uljanić

AI has been used in mededicine to develop expert systems that can assist in the diagnosis of various medical conditions. In the 1970s, two such systems were developed: Mycin and INTERNIST-1/QMR. They were built to assist in the diagnosis of various medical conditions. These early models had drawbacks which caused them to scale poorly and to generalized poorly. Diagnostics is the task of finding the cause of a phenomenon, such as a disease that is causing a symptom. Diagnostic criteria (signs, symptoms and tests) are used to determine diagnosis. Differential diagnosis looks at possible disorders and picks among possible causes. Automated diagnostics are used to establish the diagnosis by comparing the symptoms to a database of diseases and symptoms. Tests are used to help figure out what disease or condition a person has based on their signs and symptoms. Sensitivity and specificity are measures of the accuracy of a diagnostic test. The former represents the ratio of correctly identified negatives to actual negatives, while the latter represents the ratio of correctly identified positives to actual positives.

Advances in machine learning and statistics have brought better algorithms for image analysis and learning with high dimension data. AI can improve consistency and speed of diagnosis, as well as explore data which is not limited to known explanations (features learned from data). However, there also exist concerns about the reliability and reproducibility, and the potential downsides of using AI in medicine. The algorithms must be robust, fair and accountable, have checks and balances to detect errors, and have interpretable outputs. Privacy and fairness are also concerns, given that medical data is sensitive.

Module 4 reflection by Nina Uljanić

Module 4 has talked about a field I have some experience in. Implementing a NLP system using the IBM model 1 has proven to be quite straightforward, however its accuracy was not as high as one would like. It could be interesting to try to implement other models as well. I am currently taking a course on NLP, so this module was a revision and further exercise for me.

Module 5 - Diagnostic Systems

February 21, 2023

[ ]:
```
import pickle
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sn
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
```

[ ]:
```
names = 'C:/Users/hsahn/Design of AI systems/module5/wdbc.names'
table = 'C:/Users/hsahn/Design of AI systems/module5/wdbc.pkl'
```

[ ]:
```
with open(table, 'rb') as f:
    df_table_dataset = pickle.load(f)
```

[ ]:
```
df_table_dataset
```

[ ]:
```
df_table_dataset[df_table_dataset['malignant'] == 1]
```

[ ]:
```
df_table_dataset[df_table_dataset['malignant'] == 0]
```

1 Rule Based Classifier

This classifier uses the following features from the data set: 1. for cell size - perimeter_0 2. for cell shape - compactness_2 3. for cell homogenity - concave points_2 4. for cell texture - texture_2

[ ]:
```
df_table_dataset
```

[ ]:
```
df_rule_based_classifier =␣
↪df_table_dataset[['perimeter_0','compactness_2','concave points_2',␣
↪'texture_2']]
```

[ ]:
```
df_rule_based_classifier
```

[ ]:
```
X_our_model = df_rule_based_classifier.to_numpy()
```

[ ]:
```
X_our_model
```

[ ]:
```
y_our_model = df_table_dataset['malignant'].to_numpy()
```

[ ]:
```
y_our_model
```

[ ]:
```
X_our_model.shape
```

[ ]:
```
def PredictionOurModel(values, X_our_model):
    labels = []
    for i in range(X_our_model.shape[0]):
        cell_size = X_our_model[i,0]
        cell_shape = X_our_model[i,1]
        cell_homogenity = X_our_model[i,2]
        cell_texture = X_our_model[i,3]
        if (cell_size >= values[0] or cell_shape >= values[1] or␣
 ↪cell_homogenity >= values[2] or cell_texture >= values[3]):
            label = 1
        else:
            label = 0
        labels.append(label)
    return labels
```

[ ]:
```
values = [114.5, 0.4, 0.1747 , 35]
```

[ ]:
```
X = X_our_model
y = y_our_model

skf = StratifiedKFold(n_splits = 4)


accuracies = []
cf_score = []

for train_index , test_index in skf.split(X,y):
    X_train , X_test = X[train_index], X[test_index]
    y_train , y_test = y[train_index] , y[test_index]

    y_pred = PredictionOurModel(values, X_test)
    accuracy = accuracy_score(y_pred, y_test)
    accuracies.append(accuracy)

    cf_array = confusion_matrix(y_test, y_pred)
    df_cm = pd.DataFrame(cf_array, ['Benign', 'Malignant'], ['Benign',␣
 ↪'Malignant'])
    cf_score.append(cf_array)
```

```
mean_accuracy = sum(accuracies)/len(accuracies)

print('accuracy for each fold - {}'.format(accuracies))
print('Mean accuracy : {}'.format(mean_accuracy))
print("Confusion matrix:")
# print(cf_array)
print('Confusion matrix of each fold - {}'.format(cf_score))
```

[ ]:
```
plt.figure(figsize=(16,16))
counter = 0
for i in range(2):
    for j in range(2):
        df_cm = pd.DataFrame(cf_score[counter], ['Benign', 'Malignant'],␣
↪['Benign', 'Malignant'])
        plt.subplot(2, 2, counter + 1)
        sn.set(font_scale=1.5,) # Adjust font size
        sn.heatmap(df_cm, annot=True, annot_kws={"size": 18},cmap='Blues')
        plt.xlabel('Predicted values', fontsize=22)
        plt.ylabel('True values', fontsize=22)
        plt.title(('Confusion matrix for batch number ' + str(counter + 1)),␣
↪fontsize=18)
        counter = counter + 1
plt.show()
```

2 Justification for chossing the above mentioned features in our classifier

[ ]:
```
import plotly.figure_factory as ff

def plot_density(df, column):
    # Separate data by malignant status
    malignant = df_table_dataset.loc[df_table_dataset['malignant'] == 1, column]
    benign = df_table_dataset.loc[df_table_dataset['malignant'] == 0, column]

    # Create distplot with custom bin_size
    fig = ff.create_distplot([malignant, benign], ['Malignant', 'Benign'],␣
↪bin_size=1, show_hist=False)
    fig.update_layout(title_text=f'Density plot for {column.capitalize()}',␣
↪title_x=0.5, font_size=18)
    fig.show()
```

[ ]:
```
plot_density(df_table_dataset, 'perimeter_0')
```

[ ]:
```
plot_density(df_table_dataset, 'compactness_2')
```

[ ]:
```
plot_density(df_table_dataset, 'concave points_2')
```

[ ]:
```
plot_density(df_table_dataset, 'texture_2')
```

3 Random Forest Classifier

[ ]:
```
df_table_dataset
```

[ ]:
```
X_random_forest_model = df_table_dataset.drop(['id', 'malignant'], axis =␣
↪'columns')
X_random_forest_model = X_random_forest_model.to_numpy()
```

[ ]:
```
X_random_forest_model
```

[ ]:
```
y_random_forest_model = df_table_dataset['malignant'].to_numpy()
```

[ ]:
```
y_random_forest_model
```

[ ]:
```
X = X_random_forest_model
y = y_random_forest_model

skf = StratifiedKFold(n_splits = 4)

accuracies = []
cf_score = []

for train_index , test_index in skf.split(X,y):
    X_train , X_test = X[train_index], X[test_index]
    y_train , y_test = y[train_index] , y[test_index]

    clf = RandomForestClassifier(n_estimators= 100, max_depth=10)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)

    accuracy = accuracy_score(y_pred, y_test)
    accuracies.append(accuracy)

    cf_array = confusion_matrix(y_test, y_pred)
    df_cm = pd.DataFrame(cf_array, ['Benign', 'Malignant'], ['Benign',␣
↪'Malignant'])
    cf_score.append(cf_array)

mean_accuracy = sum(accuracies)/len(accuracies)

print('accuracy for each fold - {}'.format(accuracies))
print('Mean accuracy : {}'.format(mean_accuracy))
print("Confusion matrix:")
print('Confusion matrix of each fold - {}'.format(cf_score))
```

[ ]:
```
plt.figure(figsize=(16,16))
counter = 0
for i in range(2):
    for j in range(2):
        df_cm = pd.DataFrame(cf_score[counter], ['Benign', 'Malignant'],␣
 ↪['Benign', 'Malignant'])
        plt.subplot(2, 2, counter + 1)
        sn.set(font_scale=1.5,) # Adjust font size
        sn.heatmap(df_cm, annot=True, annot_kws={"size": 18},cmap='Greens')
        plt.xlabel('Predicted values', fontsize=22)
        plt.ylabel('True values', fontsize=22)
        plt.title(('Confusion matrix for batch number ' + str(counter + 1)),␣
 ↪fontsize=18)
        counter = counter + 1
plt.show()
```

4 Decision Tree

[ ]:
```
df_table_dataset
```

[ ]:
```
X_decision_tree_model = df_table_dataset.drop(['id', 'malignant'], axis =␣
 ↪'columns')
X_decision_tree_model = X_decision_tree_model.to_numpy()
```

[ ]:
```
y_decision_tree_model = df_table_dataset['malignant'].to_numpy()
```

[ ]:
```
X = X_decision_tree_model
y = y_decision_tree_model

skf = StratifiedKFold(n_splits = 4)

accuracies = []
cf_score = []

for train_index , test_index in skf.split(X,y):
    X_train , X_test = X[train_index], X[test_index]
    y_train , y_test = y[train_index] , y[test_index]

    clf = DecisionTreeClassifier(max_depth=3)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)

    accuracy = accuracy_score(y_pred, y_test)
    accuracies.append(accuracy)

    cf_array = confusion_matrix(y_test, y_pred)
```

```
    df_cm = pd.DataFrame(cf_array, ['Benign', 'Malignant'], ['Benign',␣
 ↪'Malignant'])
    cf_score.append(cf_array)

mean_accuracy = sum(accuracies)/len(accuracies)

print('accuracy for each fold - {}'.format(accuracies))
print('Mean accuracy : {}'.format(mean_accuracy))
print("Confusion matrix:")
print('Confusion matrix of each fold - {}'.format(cf_score))
```

[ ]:
```
plt.figure(figsize=(16,16))
counter = 0
for i in range(2):
    for j in range(2):
        df_cm = pd.DataFrame(cf_score[counter], ['Benign', 'Malignant'],␣
 ↪['Benign', 'Malignant'])
        plt.subplot(2, 2, counter + 1)
        sn.set(font_scale=1.5,) # Adjust font size
        sn.heatmap(df_cm, annot=True, annot_kws={"size": 18},cmap='Reds')
        plt.xlabel('Predicted values', fontsize=22)
        plt.ylabel('True values', fontsize=22)
        plt.title(('Confusion matrix for batch number ' + str(counter + 1)),␣
 ↪fontsize=18)
        counter = counter + 1
plt.show()
```