# BIOM5405/SYSC5405

## Assignment 3

## Contents

# BIOM5405/SYSC5405

## Assignment 3

# BIOM5405/SYSC5405

## Assignment 3

### List of Figures

# BIOM5405/SYSC5405

# Assignment 3

## Q1

You decide to fit a 2D Bayesian classifier to your data, where x = [T RR], COVID is the 'positive' class, and we assume that p(x|ωi ) ~ N(μi ,Σi ). Use unbiased estimators to estimate the 2D mean and covariance matrix for each class-conditional distribution. Report your two estimated mean vectors and covariance matrices.

### Solution:

Mean Healthy Vector: $\begin{bmatrix}36.7869\\21.0115\end{bmatrix}$

Mean Covid Vector: $\begin{bmatrix}39.1792\\20.9254\end{bmatrix}$

Covariance Healthy Matrix: $\begin{bmatrix}0.0698 & -0.0152\\-0.0152 & 0.1686\end{bmatrix}$

Covariance Covid Matrix: $\begin{bmatrix}2.3824 & -0.7629\\-0.7629 & 60.7494\end{bmatrix}$

### Extra 1: Verification on Covariance

Step 1: Are they symmetric – Yes

Step 2: Are the diagonals variances of there respective values – Yes

Eg: For Healthy – variance of temperature = square of STD = $(0.264226)^2 = 0.0698$

### Extra 2: Covariance representation:

Covariance Matrix of X,Y= : $\begin{bmatrix}VAR(X) & COV(X,Y)\\COV(X,Y) & VAR(Y)\end{bmatrix}$

### Extra 3: Multivariate Mean Representation:

It is always recommended to represent a multivariate vector as a single column row matrix.

### Code:

```
# Code to read the data and other unnecessary info is provided in the Appendix

# Calculate mean vectors for each class
mean_vector_healthy = np.array([data["T_healty"].mean(),
data["RR_healty"].mean()])
mean_vector_covid = np.array([data["T_covid"].mean(),
data["RR_covid"].mean()])

# Calculate covariance matrices for each class
cov_matrix_healthy = np.cov(data["T_healty"], data["RR_healty"], ddof=1)
cov_matrix_covid = np.cov(data["T_covid"], data["RR_covid"], ddof=1)
```

# BIOM5405/SYSC5405

# Assignment 3

## Q2

COVID is happily now less prevalent than it was when the data were first collected. We can now assume that, for every one person with COVID, there are 9 people without COVID in the population.

i) Use Bayes' theorem to compute the posterior probability that a patient with a temperature of 37.5 degrees and a respiration rate of 23 is healthy.

ii) Determine (analytically or through trial-and-error), what is the minimum temperature at which this patient (RR=23) will be classified as having covid.

## Solution:

Part i:

Posterior probability that the patient is healthy given T=37.5 and RR=23 = 0.8076

Part ii:

The minimum temperature after the threshold is 37.62644

The minimum possible temperature would be 0 Kelvin

The minimum possible temperature in the dataset is 34.92644

### Extra 1: Posterior Probability Plot

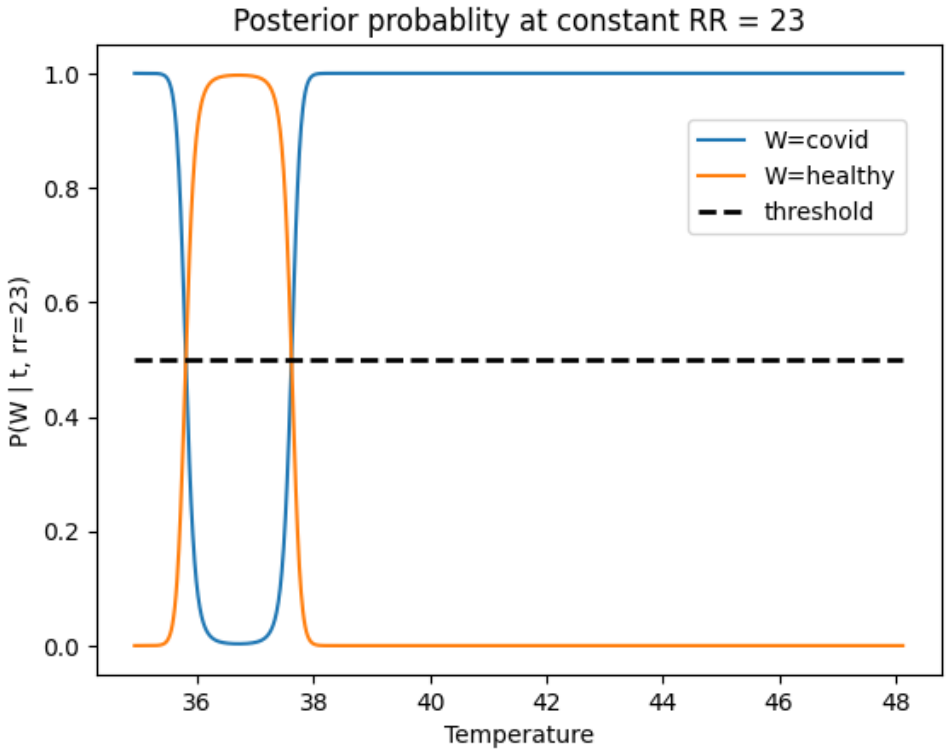


*Figure 1: Posterior Probability Function Plot for Healthy and COVID at RR=23*

# BIOM5405/SYSC5405

## Assignment 3

Based on Figure 1 we see the classifier will classify as COVID when T is less than 35.79644 and T is greater than 37.62644

### Extra 2: False assumption on distribution

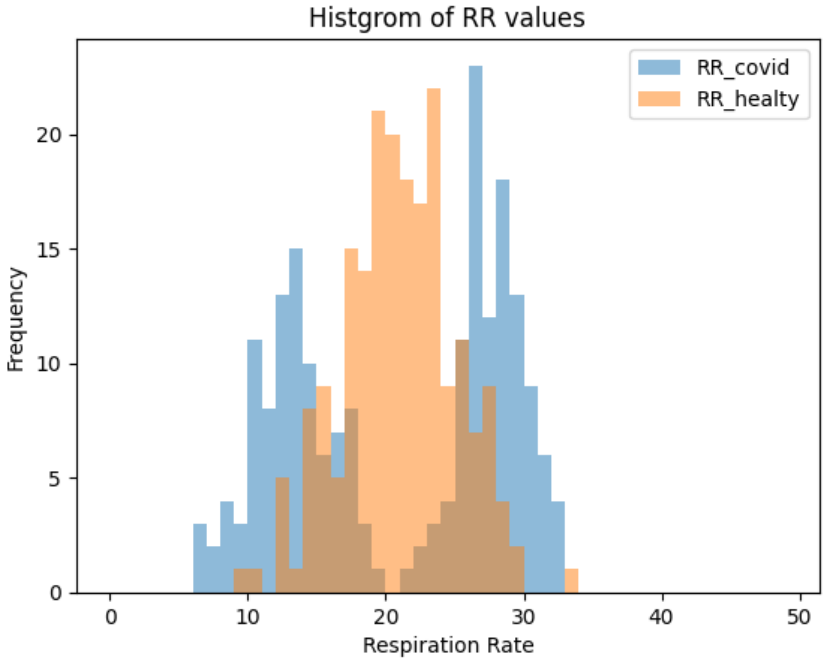


*Figure 2: Distribution of RR*

We are assuming the distribution to be normal but clearly from Figure 2, we know it to not be true.

### Code:

Part i:

```
# Code to read the data and other unnecessary info is provided in the Appendix
# Mean and Covariance values are taken from the Q1

# observation
t = 37.5
rr = 23
observation = np.array([t,rr])

# Likelihoods
likelihood_healthy = multivariate_normal(mean=mean_vector_healthy,
cov=cov_matrix_healthy).pdf(observation)
likelihood_covid = multivariate_normal(mean=mean_vector_covid,
cov=cov_matrix_covid).pdf(observation)

# Priors
prior_healthy = 0.9
```

# BIOM5405/SYSC5405

## Assignment 3

```
prior_covid = 0.1

# Evidence
w_sum_p = (likelihood_healthy * prior_healthy) + (likelihood_covid *
prior_covid)
posterior_probability = (likelihood_healthy * prior_healthy) / w_sum_p
```

Part ii:

```
# Code to read the data and other unnecessary info is provided in the Appendix
# Mean and Covariance values are taken from the Q1

# init
rr = 23
prior_covid = 0.1
prior_healthy = 0.9
t_max = max(data["T_healty"].max(), data["T_covid"].max())
t_min = min(data["T_healty"].min(), data["T_covid"].min())
t_curr = t_min
mvn_covid = multivariate_normal(mean=mean_vector_covid, cov=cov_matrix_covid)
mvn_healthy = multivariate_normal(mean=mean_vector_healthy,
cov=cov_matrix_healthy)
t_postirior_covid = []
temperatures = []

# Calculate positirier values for varying temperature values
while t_curr < t_max:
    observation = np.array([t_curr, rr])
    likelihood_covid = mvn_covid.pdf(observation)
    likelihood_healthy = mvn_healthy.pdf(observation)
    posterior_covid = likelihood_covid * prior_covid / ((likelihood_covid *
prior_covid) + (likelihood_healthy * prior_healthy))
    t_postirior_covid.append(posterior_covid)
    temperatures.append(t_curr)
    t_curr += 0.01
```

# BIOM5405/SYSC5405

# Assignment 3

## Q3

For the Bayesian classifier in Q2, compute the probability that each of the 400 patients has COVID, given their observed temperature and RR. Do not report the posterior probability for each patient. Instead, plot an ROC and a P-R curve for your classifier over these 400 patients.

- For the ROC plot, include the AUC-ROC in the title.
- For the P-R curve, include the average precision (across all recall values) in the title.

## Solution

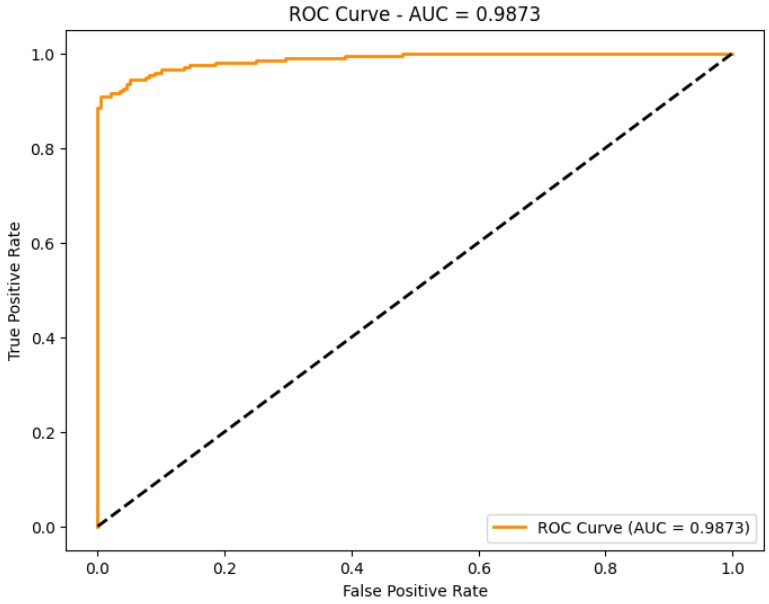


*Figure 3: ROC plot with AUC value*

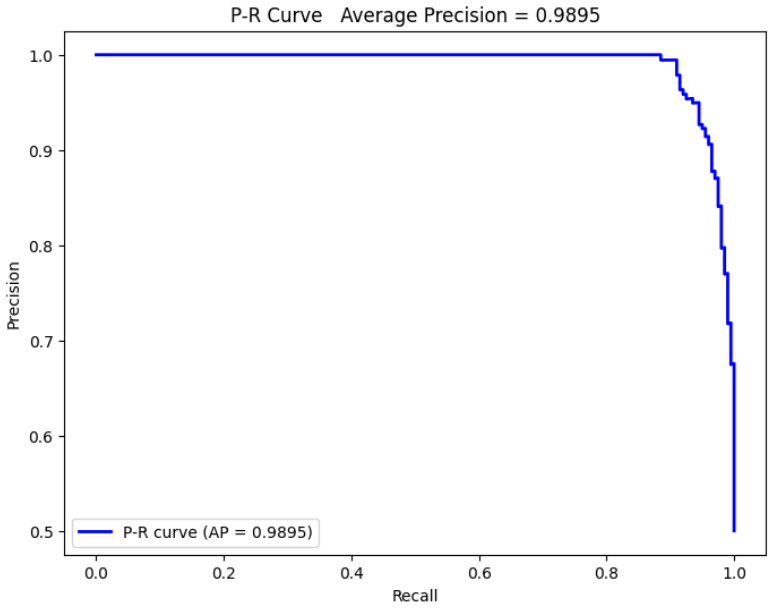


*Figure 4: P-R Curve with average precision*

# BIOM5405/SYSC5405

# Assignment 3

## Extra 1: Formula for ROC curve

$$TPR = Recall = Sensitivity = \frac{TP}{TP + FN}$$

Where,

TPR is the “True Positive Rate”,

TP is the number of correctly classified positive labels,

FN is the number of falsely classified negative labels.

Note: TPR, Recall and Sensitivity are all the same metric.

$$FPR = 1 - Specificity = \frac{FP}{TN + FP}$$

Where,

FPR is the “False Positive Rate”,

FP is the number of falsely classified positive samples,

TN is the number of correctly classified negative samples.

AUC is calculate by integrating the curve under the range [0-1] for FPR.

## Extra 2: Formula for PR curve

$$Precision = \frac{TP}{TP + FP}$$

Where,

TP is the number of correctly classified positive labels,

FP is the number of falsely classified positive labels.

$$Recall = TPR = Sensitivity = \frac{TP}{TP + FN}$$

Where,

TPR is the “True Positive Rate”,

TP is the number of correctly classified positive labels,

FN is the number of falsely classified negative labels.

# BIOM5405/SYSC5405

## Assignment 3

### Code

```
# Code to read the data and other unnecessary info is provided in the Appendix
# Mean and Covariance values are taken from the Q1

# build dataset
dataset_covid = pd.DataFrame(data[["T_covid","RR_covid"]])
dataset_covid.columns = ["t","rr"]
dataset_covid["label"] = 1

dataset_healthy = pd.DataFrame(data[["T_healty","RR_healty"]])
dataset_healthy.columns = ["t","rr"]
dataset_healthy["label"] = 0

dataset = pd.concat(
    [dataset_covid,dataset_healthy],
    axis=0
)


# make predictions
classifications = []

mvn_covid = multivariate_normal(mean=mean_vector_covid, cov=cov_matrix_covid)
mvn_healthy = multivariate_normal(mean=mean_vector_healthy,
cov=cov_matrix_healthy)

for row in dataset.iterrows():
    observation = np.array([row[1]['t'], row[1]['rr']])
    likelihood_covid = mvn_covid.pdf(observation)
    likelihood_healthy = mvn_healthy.pdf(observation)
    posterior_covid = likelihood_covid * prior_covid / ((likelihood_covid *
prior_covid) + (likelihood_healthy * prior_healthy))
    classifications.append(posterior_covid)

dataset["prediction"] = classifications

# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(dataset["label"], dataset["prediction"])
roc_auc = auc(fpr, tpr)

# Calculate P-R curve
precision, recall, thresholds_pr = precision_recall_curve(dataset["label"],
dataset["prediction"])
average_precision = average_precision_score(dataset["label"],
dataset["prediction"])
```

# BIOM5405/SYSC5405

## Assignment 3

```
# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC Curve (AUC = %0.4f)' %
roc_auc)
plt.plot([0, 1], [0, 1], color='black', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title(f'ROC Curve - AUC = {roc_auc:0.4f}')
plt.legend(loc='lower right')
plt.show()

# Plot P-R curve
plt.figure(figsize=(8, 6))
plt.plot(recall, precision, color='blue', lw=2, label='P-R curve (AP = %0.4f)'
% average_precision)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title(f'P-R Curve   Average Precision = {average_precision:0.4f}')
plt.legend(loc='lower left')
plt.show()
```

# BIOM5405/SYSC5405

# Assignment 3

## Q4

Given the high cost of false positives, you decide that your false positive rate must be below 15%.

i) What is the maximum sensitivity we can achieve?
ii) What is the maximum precision that we can achieve?
iii) Report a confusion matrix for this decision threshold.

## Solution

i) Maximum sensitivity that can be achieved: 0.975
ii) Precision:
   a. Maximum precision that can be achieved: 1.0
   b. Minimum precision that can be achieved: 0.8699551569506726
iii) Confusion matrix:

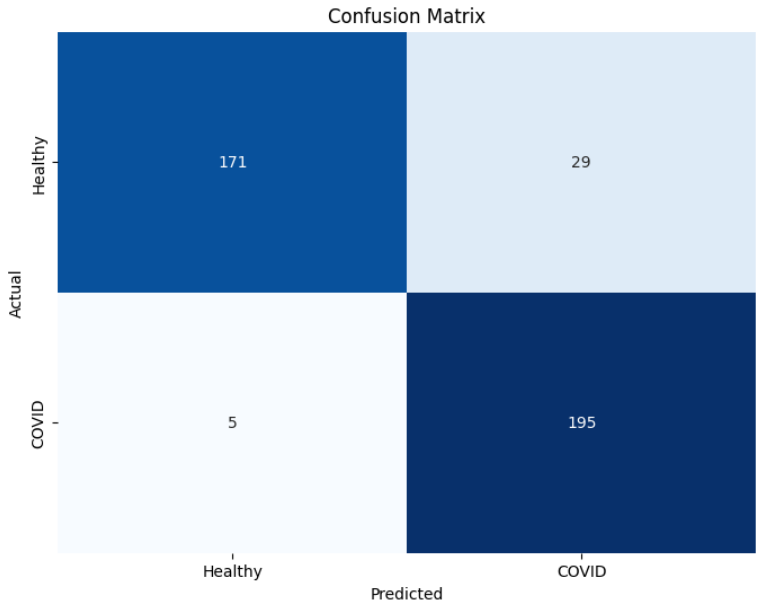


*Figure 5: Confusion matrix at threshold = 0.016224 and fpr = 0.15*

## Extra 1: Threshold to get FPR = 0.15

Threshold: 0.016224

At the above threshold the FPR is exactly 0.15

# BIOM5405/SYSC5405

## Assignment 3

### Extra 2: Plot of metrics VS the threshold

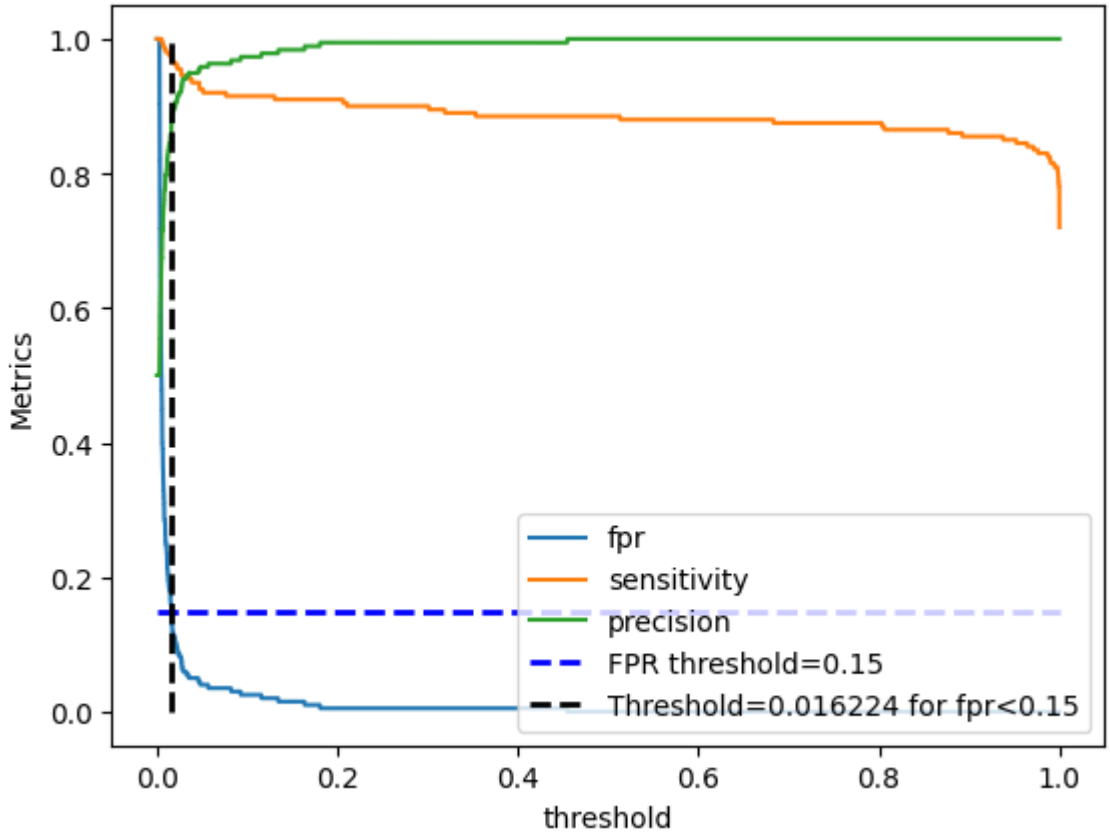


*Figure 6: Plot of all the metrics with varying values of threshold*

### Code

```
# permutate to get the metrics for varying thresholds
fpr_list = []
precison_list = []
sensitivity_list = []
confusion_matrix_list = []

for threshold in np.array(range(0,1000000,1))/1000000:
    y_pred = (dataset['prediction'] > threshold).astype(int)
    confusion = confusion_matrix(dataset['label'], y_pred)
    tn, fp, fn, tp = confusion.ravel()
    fpr = fp / (tn+fp)
    sensitivity = tp/(tp+fn)
    precision = tp/(tp+fp)
    fpr_list.append(fpr)
    sensitivity_list.append(sensitivity)
    precison_list.append(precision)
    confusion_matrix_list.append(confusion)
```

# BIOM5405/SYSC5405

## Assignment 3

```
# filter the results
metrics_df = pd.DataFrame(
    {
        "threshold": np.array(range(0,1000000,1))/1000000,
        "fpr": fpr_list,
        "sensitivity": sensitivity_list,
        "precision": precison_list,
    },
    index=range(1000000)
).set_index("threshold")

acceptable_values = metrics_df[metrics_df['fpr']<0.15]
min_threshold = min(acceptable_values.index)
max_senitivity = max(acceptable_values['sensitivity'])
max_precision = max(acceptable_values["precision"])
min_precision = min(acceptable_values["precision"])
```

```
# get the confusion matrix
y_pred = (dataset['prediction'] > min_threshold).astype(int)
confusion = confusion_matrix(dataset['label'], y_pred)

plt.figure(figsize=(8, 6))
sns.heatmap(confusion, annot=True, fmt='d', cmap='Blues', cbar=False,
xticklabels=['Healthy', 'COVID'], yticklabels=['Healthy', 'COVID'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

# BIOM5405/SYSC5405

## Assignment 3

### Q5

To account for the fact that the class imbalance in the deployment environment (1:9) is very different from the class imbalance among your 400 test samples (1:1), you decide to add 8 additional copies of each healthy patient to your test set leading to 2000 samples in total. Without 'retraining' your classifier, report the ROC and P-R curves for this new test set, along with ROC-AUC and average precision. Briefly discuss what changed, what didn't, and why. (75 words)

### Solution

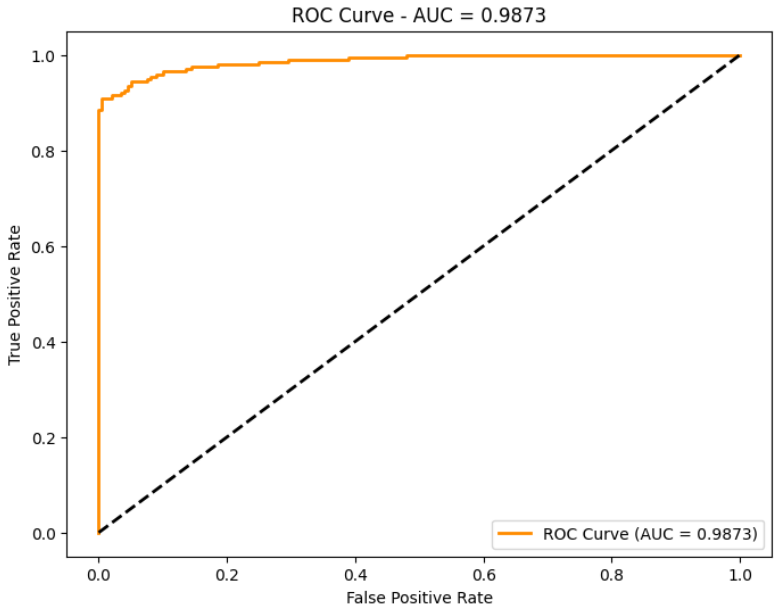


*Figure 7: ROC curve with duplicate data for healthy patients*

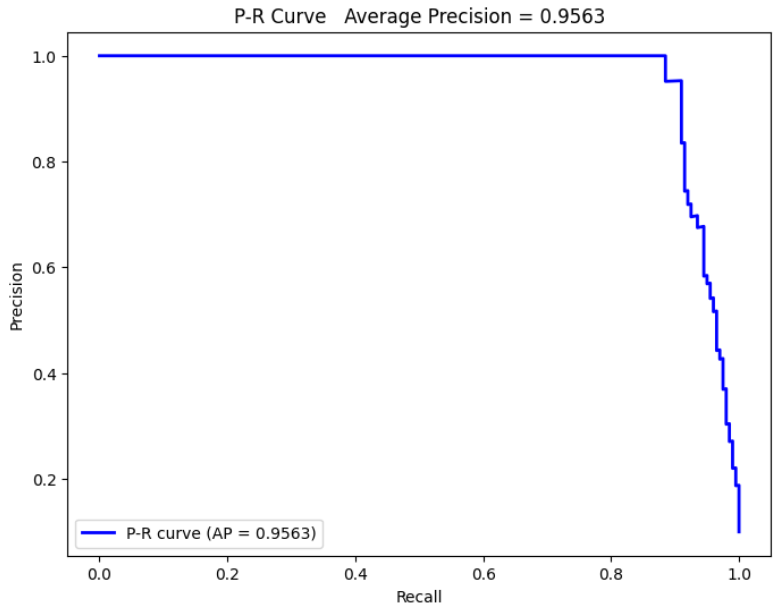


*Figure 8: P-R curve with duplicate health data*

# BIOM5405/SYSC5405

# Assignment 3

The AUC-ROC remains the same while the P-R curve and the average precision changes as we add more samples. The average precision decreased as there are more samples that will be classified incorrectly. A detailed breakdown is found in Extra 1 and Extra 2.

## Extra 1: Mathematical understanding for the change in the P-R curve

Let us consider two cases:

a) Where the ratio of covid to healthy patients is 1:1
b) Where the ratio of covid to healthy patients is 1:9

Let us now draw the confusion matrix for both cases:

| | | Predicted | |
|---|---|---|---|
| | | Covid | Healthy |
| Actual | Covid | TP | FN |
| | Healthy | FP | TN |

*Table 1: Confusion Matrix for case a*

| | | Predicted | |
|---|---|---|---|
| | | Covid | Healthy |
| Actual | Covid | TP | FN |
| | Healthy | 9FP | 9TN |

*Table 2: Confusion Matrix for case b*

The confusion matrix remains same for any given value of the threshold as the data has been duplicated. If new data were added, then there is a possibility of that the confusion matrix will not be the same.

The formula for precision and recall can be found in Q3 Extra 2.

Let us consider precision to be “y” and recall to be “x”, then we get the following values for case 2:

$y = \frac{TP}{TP+9FP}$ and $x = \frac{TP}{TP+FN}$

Therefore, we will get a shift in the y-axis values changing the new plot.

We see from this that the x value changes. Therefore, there is a change in the plot.

Also the average precision values will also change as there is a change in the denominator for precision.

## Extra 2: Mathematical understanding for the change in the ROC curve:

Let us consider two cases:

a) Where the ratio of covid to healthy patients is 1:1
b) Where the ratio of covid to healthy patients is 1:9

Let us now draw the confusion matrix for both cases:

# BIOM5405/SYSC5405

## Assignment 3

| | | Predicted | |
|---|---|---|---|
| | | Covid | Healthy |
| Actual | Covid | TP | FN |
| | Healthy | FP | TN |

*Table 3: Confusion Matrix for case a*

| | | Predicted | |
|---|---|---|---|
| | | Covid | Healthy |
| Actual | Covid | TP | FN |
| | Healthy | 9FP | 9TN |

*Table 4: Confusion Matrix for case b*

The confusion matrix remains same for any given value of the threshold as the data has been duplicated. If new data was added, then there is a possibility of that the confusion matrix will not be the same.

The formula for FPR and TPR can be found in Q3 Extra 1.

Let us consider TPR to be "y" and FPR to be "x", then we get the following values for case 2:

$y = \frac{TP}{TP+FN}$ and $x = \frac{9FP}{9TP+9FN} = \frac{9\times FP}{9\times(TP+FN)} = \frac{FP}{TP+FN}$

In this case the x and y values do not change. Therefore, it remains the same.

### Code

```
# Code to read the data and other unnecessary info is provided in the Appendix
# Mean and Covariance values are taken from the Q1 dataset values are taken
from Q3

# create new dataset
copy_healthy = 9
copy_covid = 1
dataset_list = [dataset_healthy for _ in range(copy_healthy)] + [dataset_covid
for _ in range(copy_covid)]
new_datset = pd.concat(dataset_list)

# Make predictions
classifications = []

mvn_covid = multivariate_normal(mean=mean_vector_covid, cov=cov_matrix_covid)
mvn_healthy = multivariate_normal(mean=mean_vector_healthy,
cov=cov_matrix_healthy)

for row in new_datset.iterrows():
    observation = np.array([row[1]['t'], row[1]['rr']])
    likelihood_covid = mvn_covid.pdf(observation)
    likelihood_healthy = mvn_healthy.pdf(observation)
```

# BIOM5405/SYSC5405

## Assignment 3

```
    posterior_covid = likelihood_covid * prior_covid / ((likelihood_covid *
prior_covid) + (likelihood_healthy * prior_healthy))
    classifications.append(posterior_covid)

new_datset["prediction"] = classifications

# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(new_datset["label"],
new_datset["prediction"])
roc_auc = auc(fpr, tpr)

# Calculate P-R curve
precision, recall, thresholds_pr = precision_recall_curve(new_datset["label"],
new_datset["prediction"])
average_precision = average_precision_score(new_datset["label"],
new_datset["prediction"])

# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC Curve (AUC = %0.4f)' %
roc_auc)
plt.plot([0, 1], [0, 1], color='black', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title(f'ROC Curve - AUC = {roc_auc:0.4f}')
plt.legend(loc='lower right')
plt.show()

# Plot P-R curve
plt.figure(figsize=(8, 6))
plt.plot(recall, precision, color='blue', lw=2, label='P-R curve (AP = %0.4f)'
% average_precision)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title(f'P-R Curve   Average Precision = {average_precision:0.4f}')
plt.legend(loc='lower left')
plt.show()
```

# BIOM5405/SYSC5405

# Assignment 3

## Q6

Passengers who have been granted access to the buffet but were actually sick with COVID may cause an epidemic aboard the ship. Which performance metric reflects the chance that a COVID-positive person was permitted to use the buffet? (15 words, plus equation for performance metric)

### Solution

In a confusion matrix, the False Negative value represents the value. Considering this “False Negative Rate” (FNR) is the metric to use.

$$FNR = \frac{FN}{TP+FN}$$

### Extra 1: Other metrics to consider.

One can also use Sensitivity (recall) as it is “1-FNR”. Let us keep in mind that high sensitivity will mean that we are letting in low number of falsely misclassified covid patients in. While a high FNR will mean that we are letting in a high number of misclassified covid patients in.

### Extra 2: Plot for FNR where covid patients is equal to non-covid patients.

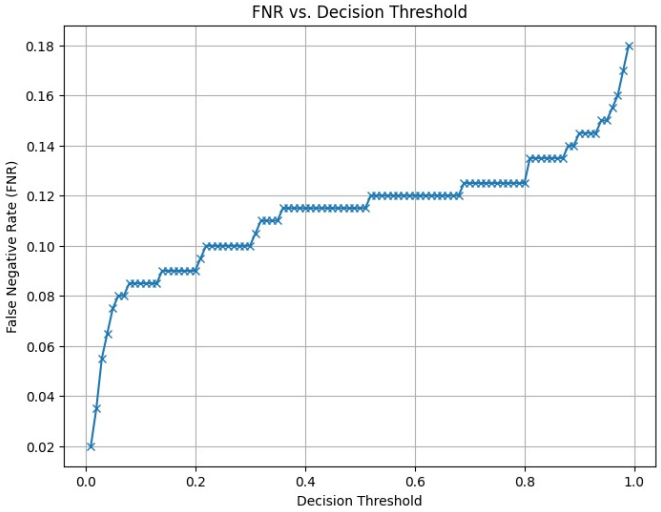


*Figure 9: FNR vs decision threshold*

### Code:

There is no code needed for this question.

# BIOM5405/SYSC5405

# Assignment 3

## Q7

We will now use a K-nearest-neighbour classifier to classify all passengers in the original 400-patient data set (ignore prior information). Report the apparent error rate for K-NN classifiers with K={1,5,15,25}. Which value of the hyperparameter, K, performs best and why? (50 words; reminder that you can use an existing K-NN library here…)

### Solution

K = 1: Apparent Error Rate = 0.0000

K = 5: Apparent Error Rate = 0.0600

K = 15: Apparent Error Rate = 0.0775

K = 25: Apparent Error Rate = 0.0850

The best performing K is K = 1.

When K=1, we are testing the classifier on the same data it trained on. The classifier has already memorised the right value and can give 100% correct classification no matter what.

This K=1 is a very wrong way to test when the test and train data is the same.

In cases where K>1, the input is comes from the neighbouring values as well, leading to some misclassification near the true decision boundary. And hence a lower error rate.

### Extra 1: Visual understanding on for case K=1

We know the train and test data are the same so let is take 1 arbitrary point sample and get an understanding.

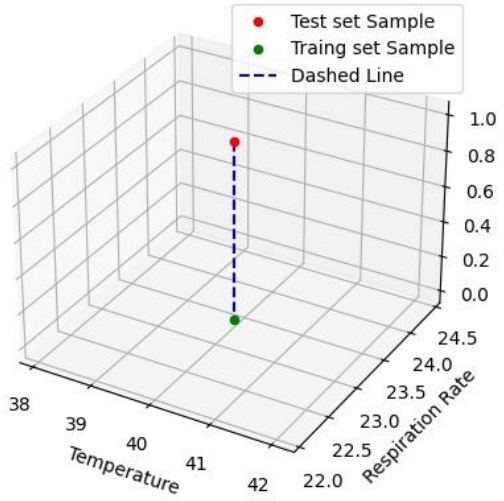


*Figure 10: Understanding of K=1 case*

The nearest point to any test set sample will be the sample point in the train set. Since the train set is already correctly classified, we will always get the correct values.

# BIOM5405/SYSC5405

## Assignment 3

### Code

```
# Code to read the data and other unnecessary info is provided in the Appendix
# Dataset values are taken from Q3

# init
k_values = [1, 5, 15, 25]
error_rates = []
knns = []

# Create x, y
x = dataset[['t','rr']].to_numpy()
y = dataset['label'].to_numpy()

# train knns
for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(x, y)
    y_pred = knn.predict(x)
    accuracy = accuracy_score(y, y_pred)
    error_rate = 1 - accuracy
    error_rates.append(error_rate)
    knns.append(knn)
```

# BIOM5405/SYSC5405

# Assignment 3

## Q8

Discuss how, in this assignment, we have committed both methodological errors of i) “Testing

on the training set” and ii) “Training on the testing set”.

### Solution

In this assignment, almost all questions have either or both of the problems mentioned.

Q1: We are only training and reporting the mean and covariance and not doing any testing, hence, this can be ignored.

Q2: In this question we are inferring meaning from the built classifier, hence the two still dont qualify as errors yet.

Q3: We are now calculating metrics and thus testing on train test .

Q4: Here we are again trying to infer meaning so either should not apply. But there is a possibility that one can argue that this is an example of testing in train set.

Q5: Here we trained the data on the test set as the data was duplicated but we are also testing on the train set.

Q6: Not applicable here as it is asking which metric to use.

Q7: In this question we are training on the test set. One can also argue that if we are testing it the metrics we obtain are artificially inflated so we are testing on the train set as well.

### Extra 1: Understanding

Testing on the Training set:

Testing on a training set will typically yield artificially high performance scores because the model has already seen this data during training.

Training on the Test set:

Training on a test set can lead to overfitting, where the model memorizes the test set data rather than learning general patterns. As a result, the model may perform well on the test set but poorly on unseen data.

# BIOM5405/SYSC5405

## Assignment 3

### Appendix:

#### Unimportant code:

```
from sklearn.metrics import roc_curve, auc, precision_recall_curve,
average_precision_score, accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from scipy.stats import multivariate_normal
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
```

```
# Load the data from the CSV file
columns=["T_healty", "T_covid", "RR_healty", "RR_covid"]
data = pd.read_csv("../data/A2Q2.csv",header=None)
data.columns = columns
```