Contents

[List of Figures 3](#_Toc147963929)

[Q1 4](#_Toc147963930)

[Solution 4](#_Toc147963931)

[Part 1: Building Contingency Table 4](#_Toc147963932)

[Extra 1: Calculating the Row Marginals 4](#_Toc147963933)

[Part 2: χ2 test 4](#_Toc147963934)

[Extra 2: χ2 formula 5](#_Toc147963935)

[Extra 3: Degrees of freedom 5](#_Toc147963936)

[Extra 4: χ2 Table 5](#_Toc147963937)

[Extra 5: Cell Proportion Chart 5](#_Toc147963938)

[Q2 6](#_Toc147963939)

[Solution 6](#_Toc147963940)

[Extra 1: Box Plots 6](#_Toc147963941)

[Extra 2: Conclusion 6](#_Toc147963942)

[Extra 3: Quartile Ranges 7](#_Toc147963943)

[Q3 8](#_Toc147963944)

[Solution 8](#_Toc147963945)

[Extra 1: Real Mean 8](#_Toc147963946)

[Extra 2: Visual Representation 8](#_Toc147963947)

[Q4 9](#_Toc147963948)

[Solution: 9](#_Toc147963949)

[Extra 1: Validation 9](#_Toc147963950)

[Extra 2: Box plot: 9](#_Toc147963951)

[Extra 3: Plots 10](#_Toc147963952)

[Q5 11](#_Toc147963953)

[Solution: 11](#_Toc147963954)

[Extra 1: observed deference in means 11](#_Toc147963955)

[Extra 2: Hypothesis 11](#_Toc147963956)

[Extra 3: Deep Dive 11](#_Toc147963957)

[Q6 12](#_Toc147963958)

[Solution: 12](#_Toc147963959)

[Extra 1: Comparison with additional classifiers 12](#_Toc147963960)

[Extra 2: Reasoning for Simple classifier perming high 13](#_Toc147963961)

[Extra 3: Simple Classifier Prediction Plot 13](#_Toc147963962)

[Code: 14](#_Toc147963963)

[References 23](#_Toc147963964)

# List of Figures

[Figure 1: Chi-Square Distribution Table 5](file:///C:\Users\a_mut\Desktop\Carleton\Courses\Courses%2023%20Fall\SYSC%205405%20Pattern%20Classification%20and%20Experiment%20Design\Assignment\Assignment%202\SYSC5405_-_Assignment_2.docx#_Toc147963965)

[Figure 2: Cell proportion for T\_cat and RR\_ord 5](#_Toc147963966)

[Figure 3: Box plot for T\_healthy and 10% trimmed T\_healthy. Outliers are depicted with black circles. 6](#_Toc147963967)

[Figure 4: Histogram of Temperature of Healthy people along with means 8](#_Toc147963968)

[Figure 5: Box plot creation as illustrated by (Wikipedia) 9](#_Toc147963969)

[Figure 6: Box Plot for RR\_combined 10](#_Toc147963970)

[Figure 7: Histogram of RR\_combined with bin count = 20 10](#_Toc147963971)

[Figure 8: Histogram of RR\_combined with bin count = 30 10](#_Toc147963972)

[Figure 9: Histogram for Respiration Rate for covid and healthy patients with means marked 11](#_Toc147963973)

[Figure 10: ROC for different classifiers 12](#_Toc147963974)

[Figure 11: Histogram of Temperature of Covid an Healthy people 13](#_Toc147963975)

[Figure 12: Simple Classifier predictions 13](#_Toc147963976)

# Q1

Let’s focus on our healthy patients for this question. Create a categorical version of the temperature feature data using the rule:

if T\_healthy <= 36.8, T\_cat = t-normal; else, T\_cat = t-fever.

Now create an ordinal version of the respiration rate feature data using the rule:

if RR\_healthy < 19.0, RR\_ord = RR-low;

else if RR\_healthy < 23.0, RR\_ord = RR-med;

else RR\_ord = RR-high.

Create a contingency table for your new data and use a χ2 test to check if t\_cat is significantly correlated with RR\_ord. Report your null hypothesis H0 (~15 words), your alternate hypothesis H1, your χ2 value, your degrees of freedom, your p-value, and your conclusion (~50 words).

## Solution

### Part 1: Building Contingency Table

Table 1 and 2 is built using the information presented in (Green, 2023). They provide the information on the contingency tables needed for the χ2 test.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T\_cat | Outcome=RR-high | Outcome=RR-low | Outcome=RR-med | Totals |
| t-fever | 30 | 33 | 31 | 94 |
| t-normal | 35 | 26 | 45 | 106 |
| Totals | 65 | 59 | 76 | 200 |

Table : Contingency Table (fo): Joint distribution of Outcomes (Respiration Rate) and Temperature

#### Extra 1: Calculating the Row Marginals

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T\_cat | Outcome=RR-high | Outcome=RR-low | Outcome=RR-med | Totals |
| t-fever | 31.91 % | 35.11 % | 32.98 % | 94 |
| t-normal | 33.02 % | 24.53 % | 42.45 % | 106 |
| Totals | 65 | 59 | 76 | 200 |

Table : Row Marginal Counts: Distribution in Table 1 expressed as percentages.

### Part 2: χ2 test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T\_cat | Outcome=RR-high | Outcome=RR-low | Outcome=RR-med | Totals |
| t-fever | 30.55 | 27.73 | 35.72 | 94 |
| t-normal | 34.45 | 31.27 | 40.28 | 106 |
| Totals | 65 | 59 | 76 | 200 |

Table : Expected frequencies (fe) of Table 1

***H0*: The variables T\_cat and RR\_ord are independent of each other.**

***H1*: The variables T\_cat and RR\_ord are dependent on each other.**

***χ2* = 3.0851**

***p-value* = 0.21**

***degrees of freedom* = 2**

***Conclusion*: Failed to reject null hypothesis (H0) for significance level (alpha) 0.05**

#### Extra 2: χ2 formula

(1)

χ2 value is calculate using values in Table 1 (fo) and Table 3 (fe) inequation (1).

#### Extra 3: Degrees of freedom

Degrees of freedom = (number of rows – 1) x (number of columns – 1)

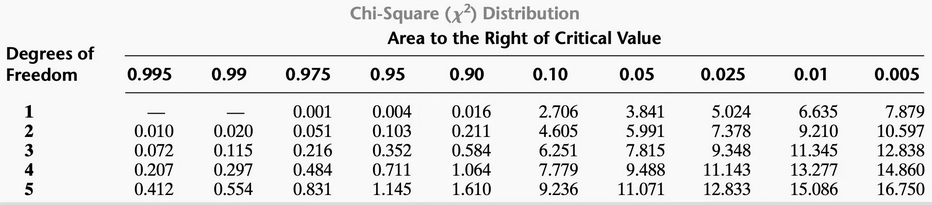
= (2-1) x (3-1)

= 1 x 2

= 2

#### Extra 4: χ2 Table

Figure : Chi-Square Distribution Table



We know the degrees of freedom is 2. The significance level (alpha) is not specified in the question. Based on the p-value calculated from χ2 (0.21) we will fail to reject the null hypothesis (H0) as long as alpha < p-value. A good reference to understand what is going on is (Tutor, 2019).

#### Extra 5: Cell Proportion Chart

From figure 2 we can see that there no change in the proportion of values in a category for different values of RR\_ord and T\_cat. This means that there very little or no correlation among the two.

Figure : Cell proportion for T\_cat and RR\_ord

# Q2

Compute the inter-quartile range and the “10% trimmed mean” of T\_healthy. (10% means dropping the top and bottom 5% of samples)

## Solution

**True Inter Quartile Range: 0.3629**

**True Mean: 36.7869**

**10% trimmed Inter Quartile range: 0.3158**

**10% trimmed mean: 36.7863**

### Extra 1: Box Plots

A diagram of a boxplots

Description automatically generated

Figure : Box plot for T\_healthy and 10% trimmed T\_healthy. Outliers are depicted with black circles.

### Extra 2: Conclusion

It can be assumed that, since the mean did not change, after the 10% trim, that there were in fact no outliers in the data. This assumption would be wrong, as we see from the Figure 3 that there was 1 outlier in the data shown in the T\_healty box plot.

### Extra 3: Quartile Ranges

Table 4 lists the quartile ranges. For 10% dropped mean.



Table : Quartile ranges

# Q3

Using bootstrapping, compute the 90% confidence interval of the “10% trimmed mean” of T\_healthy. Follow Procedure 5.6 from Cohen’s text:

1) Construct a distribution from K bootstrap samples for a statistic u; \*

2) Sort the values in the distribution

3) The lower bound of the 90% confidence interval is the (K\*0.05)th value, the upper bound is the (K\*0.95)th value in the sorted distribution.

\*Here, u is the observed trimmed mean and a bootstrap sample will consist of 200 samples drawn with replacement from T\_healthy.

## Solution

**Lower confidence interval: 36.75491145**

**Upper confidence interval: 36.820047422222224**

### Extra 1: Real Mean

Real mean: 36.78693378

### Extra 2: Visual Representation

A graph with numbers and a line

Description automatically generated with medium confidence

Figure : Histogram of Temperature of Healthy people along with means

# Q4

Examine the RR feature. Combine the RR feature data for both classes to create RR\_combined. Do the RR\_combined feature data contain outliers? Describe how you tested this and what conclusions you drew. How did the mean and median of RR\_combined change with the outliers (if any) removed? (50 words + calculations)

## Solution:

***Ans*: no outliers**

***Technique*: Box plots (Figure 6)**

***How it works*: First IRQ (Inter Quartile Range) is calculated which is upper value of Quartile 3 – lower value of Quartile 1. Then the upper and lower whisker limits are calculated which are 1.5 times the IQR. Any values that don’t fall in the whisker limits are considered as outliers. More details in** (Yi)

### Extra 1: Validation

Looking at the histogram in Figure 7 and 8. We see that ther are no sudden peaks at both left and right ends. Additionally, the data is also not spread too wide.

### Extra 2: Box plot:

Figure blow from (Wikipedia) depicts how the box plots are create.

A screenshot of a computer screen

Description automatically generated

Figure : Box plot creation as illustrated by (Wikipedia)

### Extra 3: Plots

**A graph of respiration rate

Description automatically generated**

Figure : Box Plot for RR\_combined

Figure 6 illustrates the box plot for RR\_combined (RR\_healthy and RR\_covid). The green triangle in the box denotes the mean while the green line denotes the median We see that the mean and median are similar.

A graph of a graph

Description automatically generated

Figure : Histogram of RR\_combined with bin count = 20

A graph of a graph

Description automatically generated

Figure : Histogram of RR\_combined with bin count = 30

The data for RR\_combined in tri-modal. More details in Q5 - Extra 3.

# Q5

Using randomization (or permutation), test whether RR\_covid has significantly greater mean than RR\_healthy. Briefly describe how you did this. What p-value did you obtain? What conclusion do you draw? (50 words)

## Solution:

***P-value*: 0.55565**

***Conclusion*: Significant evidence DOES NOT exists that there is a greater difference in mean for RR\_covid and RR\_healthy**

**Descreption: Following the information presented in “Slide 65” in “SYSC5405-Slides-04-ClassificationAccuracy” the test was conducted. A further detailed explanation is found in the code.**

### Extra 1: observed deference in means

Observed mean difference: -0.08615704999999707 (RR\_covid - RR\_healthy)

### Extra 2: Hypothesis

H0: There is no significant difference in means between RR\_covid and RR\_healthy

H1: There is significant evidence that RR\_covid has a greater mean than RR\_healthy.

### Extra 3: Deep Dive

A graph of a graph

Description automatically generated with medium confidence

Figure : Histogram for Respiration Rate for covid and healthy patients with means marked

The data for Covid is bi-modal. When the two modes are separated, there is a significant difference in mean, but when the two covid mode are combined the mean is similar to healthy patients.

# Q6

Let’s use temperature alone to create a simple classifier. Plot an ROC curve for temperature Assume that T\_covid samples actually have class = +1 and T\_healthy samples actually have class = 0. Our classifier will apply a tunable threshold to determine whether each sample should be predicted to have class 0 (healthy) or class 1 (covid). Report the AUC value in the title of the plot.

## Solution:

Figure 10 illustrates the ROC and AUC values of the simple classifier (Normalized Temp).  
Some additional work is also done for the question reported in the extra sections.

A graph of a number of different colored lines

Description automatically generated with medium confidence

Figure : ROC for different classifiers

### Extra 1: Comparison with additional classifiers

It can be noted that simple classifier had the highest AUC score along with Logistic regression of 0.9665.

Note: Figure 10 reports the AUC value for normalized data in the title.  
Additionally two more classifiers Logistic Regression and SVM have been trained to compare the classification power of the simple classifier. The respective AUC values are presented in the legend on the bottom right.

### Extra 2: Reasoning for Simple classifier perming high

A graph of a temperature

Description automatically generated

Figure : Histogram of Temperature of Covid an Healthy people

Since the variance of the “healthy” patients is small and at the near start of data a simple “normalization” classifier worked to separate the data.  
  
Note: In most (complex) cases this should never happen.

### Extra 3: Simple Classifier Prediction Plot

A graph showing the temperature of a temperature

Description automatically generated

Figure : Simple Classifier predictions

# Code:

# %%

import pandas as pd

import numpy as np

import math

import matplotlib.pyplot as plt

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import roc\_curve, roc\_auc\_score, auc

from scipy.stats import chi2\_contingency

data\_path = "../data/A2Q2.csv"

columns = ['T\_healty', 'T\_covid', 'RR\_healty', 'RR\_covid']

dataset = pd.read\_csv(data\_path, header=None)

dataset.columns = columns # set the headers

dataset.tail()

# # Q1

healthy\_patients\_df = pd.DataFrame()

# create a Temp Category

healthy\_patients\_df['T\_cat'] = dataset['T\_healty'].apply(

    lambda temp: "t-normal" if temp <= 36.8 else "t-fever",

    )

# Create a Resperation Category

healthy\_patients\_df['RR\_ord'] = dataset['RR\_healty'].apply(

    lambda rr: "RR-low" if rr < 19.0 else "RR-med" if rr < 23.0 else "RR-high"

    )

healthy\_patients\_df.head()

# %%

# Create a Contengency Table

contangency\_table\_df = healthy\_patients\_df[['T\_cat', 'RR\_ord']].value\_counts().reset\_index().pivot(index='T\_cat', columns='RR\_ord', values='count')

contangency\_table\_df

# %%

# Create the Expected Frequency Table:

temp\_totals = dict(contangency\_table\_df.sum(axis=1))

rr\_totals = dict(contangency\_table\_df.sum(axis=0))

total = sum(temp\_totals.values())

assert total == sum(rr\_totals.values()), f"Totals are not correct"

rr\_exp = {}

for rr in rr\_totals:

    t\_exp = {}

    for t in temp\_totals:

        freq\_exp = temp\_totals[t] \* rr\_totals[rr] / total

        t\_exp[t] = freq\_exp

    rr\_exp[rr] = t\_exp

exp\_frequencies\_df = pd.DataFrame(rr\_exp)

exp\_frequencies\_df

# %%

# Chi Square test

chi\_square = (((contangency\_table\_df - exp\_frequencies\_df)\*\*2) / exp\_frequencies\_df).sum().sum()

chi\_square

# %%

# validation

statistic, pvalue, dof, exp\_f = chi2\_contingency(contangency\_table\_df.values)

print(f"Chi-Square Value = {statistic}")

print(f"p-value = {pvalue}")

print(f"Degree of Freedom = {dof}")

print(f"Expected Frequencies = {exp\_f}")

# %% [markdown]

# do not reject null hypothess for significance level 0.05

# %% [markdown]

# # Q2

# %%

iqr = dataset["T\_healty"].describe()["75%"] - dataset["T\_healty"].describe()["25%"]

trimed\_t\_heathy = dataset["T\_healty"].sort\_values().reset\_index().iloc[10:190]

trimmed\_iqr = trimed\_t\_heathy["T\_healty"].describe()["75%"] - trimed\_t\_heathy["T\_healty"].describe()["25%"]

mean\_trimed = trimed\_t\_heathy["T\_healty"].mean()

real\_mean = dataset["T\_healty"].describe()["mean"]

print(f"Inter Quyartile Range: {iqr}")

print(f"Real mean: {real\_mean}")

print(f"Trimmed 10% Inter Quyartile Range: {trimmed\_iqr}")

print(f"Trimmed 10% mean: {mean\_trimed}")

# %%

t\_cat\_df = pd.concat([dataset["T\_healty"],trimed\_t\_heathy[["T\_healty"]]],axis=1)

t\_cat\_df.columns = ["T\_healty", "T\_healty\_10%\_trimmed"]

t\_cat\_df.plot.box(

    xlabel="Temperature",

    title="T\_healty Boxplots"

)

# %%

t\_healthy\_list = dataset["T\_healty"].values.tolist() # geta list of data

t\_healthy\_list.sort() # sort the data

len\_t\_health = len(t\_healthy\_list) # get the length of data

len\_of\_quartile = int(math.floor(len\_t\_health/4)) # get quartile size

quartiles = [

    t\_healthy\_list[len\_of\_quartile \* quartile: len\_of\_quartile \* (quartile + 1)]

    for quartile in range(4)

]

# print Quartile Ranges

for index, quartile in enumerate(quartiles):

    print(f"Quartile {index+1}: Start({quartile[0]}) End({quartile[-1]})")

# %%

# verification

dataset["T\_healty"].describe()

# %%

dataset["T\_healty"].plot.hist(bins=20)

# %%

min\_cutoff\_index = int(math.floor(len\_t\_health \* 0.05))

max\_cutoff\_index = len\_t\_health - min\_cutoff\_index

t\_healthy\_list\_10\_trimmed = t\_healthy\_list[min\_cutoff\_index:max\_cutoff\_index]

sum(t\_healthy\_list\_10\_trimmed)/len(t\_healthy\_list\_10\_trimmed)

# %% [markdown]

# # Q3

# %%

# init

K = 200000

bootstrap\_means = []

t\_healthy = dataset['T\_healty'].values

real\_mean = t\_healthy.mean()

len\_t\_health = len(t\_healthy)

lower\_10\_bound  = int(math.floor(len\_t\_health\*0.05))

upper\_10\_bound = len\_t\_health - lower\_10\_bound

K\_lower\_confidence\_bound = int(math.floor(K\*0.05))

K\_upper\_confidence\_bound = K - K\_lower\_confidence\_bound

# Run Bootstrapping

for \_ in range(K):

    # get bootstrap samples with replacement

    bootstrap\_sample = np.random.choice(

        t\_healthy,

        size=len\_t\_health,

        replace= True)

    # get 10% trimmed mean

    trimmed\_mean = bootstrap\_sample[lower\_10\_bound:upper\_10\_bound].mean()

    # add sampled mean

    bootstrap\_means.append(trimmed\_mean)

# sort the data

bootstrap\_means.sort()

# calculate the confidence intervals

lower\_confidence\_interval = bootstrap\_means[K\_lower\_confidence\_bound]

upper\_confidence\_interval = bootstrap\_means[K\_upper\_confidence\_bound]

print(f"Lower confidence interval: {lower\_confidence\_interval}")

print(f"Upper confidence interval: {upper\_confidence\_interval}")

print(f"------------------------------------------------------")

print(f"actual mean: {real\_mean}")

# %%

ax = dataset['T\_healty'].plot.hist(

    bins=20,

    histtype=u'step',

    label="t\_healthy",

    title="Distribution of T\_Healthy with 90% mean conficence bounds",

    xlabel="Temperature"

)

plt.plot([real\_mean, real\_mean], [real\_mean, 0], color='black', lw=2, linestyle='--', label=f'Real mean: {real\_mean:.3f}')

plt.plot([lower\_confidence\_interval, lower\_confidence\_interval], [lower\_confidence\_interval, 0], color='darkred', lw=2, linestyle='--', label=f'Lower 90% confidence inteval: {lower\_confidence\_interval:.3f}')

plt.plot([upper\_confidence\_interval, upper\_confidence\_interval], [upper\_confidence\_interval, 0], color='red', lw=2, linestyle='--', label=f'Upper 90% confidence inteval:{upper\_confidence\_interval:.3f}')

plt.legend(loc='upper right')

# %% [markdown]

# # Q4

# %%

# build combined features

RR\_combined = pd.DataFrame()

RR\_combined["RR\_combined"] = pd.concat(

    [dataset["RR\_covid"], dataset["RR\_healty"]],

    axis=0)

# get dataset stats

stats = RR\_combined.describe()

min\_val = stats.loc['min'].values[0]

max\_val = stats.loc['max'].values[0]

q3 = stats.loc['75%'].values[0]

q1 = stats.loc['25%'].values[0]

iqr = q3 - q1

upper\_wisker\_limit = q3 + (iqr \* 1.5)

lower\_wisker\_limit = q1 - (iqr \* 1.5)

print(f"""min: {min\_val:0.4f} --- lower\_limit: {lower\_wisker\_limit:.4f} ==> {"no outliers" if min\_val > lower\_wisker\_limit else "outliers exists"}""" )

print(f"""max: {max\_val:0.4f} --- upper\_limit: {upper\_wisker\_limit:.4f} ==> {"no outliers" if max\_val < upper\_wisker\_limit else "outliers exists"}""" )

# %%

RR\_combined.plot.box(

    ylabel="Respiration Rate",

    title="Box Plot for RR\_combined",

    figsize=(2,5),

    showmeans=True,

    grid=True,

    legend=True

)

# %%

RR\_combined.plot.hist(

    bins=20,

    xlabel="Respiration Rate",

    title="Histogram with Bin Count = 20"

)

RR\_combined.plot.hist(

    bins=30,

    xlabel="Respiration Rate",

    title="Histogram with Bin Count = 30"

)

# %% [markdown]

# # Q5

# %%

# init

significance\_level\_p\_val = 0.05

num\_permutations = 100000

permuation\_mean\_diff\_samples = []

rr\_covid = dataset["RR\_covid"].values

rr\_healthy = dataset["RR\_healty"].values

rr\_covid\_count = len(rr\_covid)

rr\_healthy\_count = len(rr\_healthy)

# calculate observed mean

rr\_covid\_mean = rr\_covid.mean()

rr\_healthy\_mean = rr\_healthy.mean()

real\_mean\_diff = rr\_covid\_mean - rr\_healthy\_mean

# Concatenate data

rr\_conbined = np.concatenate([rr\_covid, rr\_healthy])

# Run Permuation

for \_ in range(num\_permutations):

    # shuffle data

    np.random.shuffle(rr\_conbined)

    # sample the data without replacements

    sampled\_rr\_covid = rr\_conbined[:rr\_covid\_count]

    sampled\_rr\_healthy = rr\_conbined[rr\_covid\_count:]

    # Calcuate test statstatics

    sampled\_rr\_covid\_mean = sampled\_rr\_covid.mean()

    sampled\_rr\_healthy\_mean = sampled\_rr\_healthy.mean()

    sampled\_mean\_diffrence = sampled\_rr\_covid\_mean - rr\_healthy\_mean

    # add the observation

    permuation\_mean\_diff\_samples.append(sampled\_mean\_diffrence)

# calculate p-value

p\_val = sum([sample >= real\_mean\_diff for sample in permuation\_mean\_diff\_samples])/num\_permutations

print(f"Observed mean diffrenc: {real\_mean\_diff}")

print(f"P-value: {p\_val}")

# Conclusion

if p\_val < significance\_level\_p\_val:

    print("Significant evidence exists that there is a greater deiffrence in mean for RR\_covid and RR\_healthy")

else:

    print("Significant evidence DOES NOT exists that there is a greater deiffrence in mean for RR\_covid and RR\_healthy")

# %%

dataset[["RR\_covid","RR\_healty"]].plot.hist(

    bins=40,

    color=[(1,0,0,0.5),(0,1,0,0.5)],

    title="Histogram of Respiration Rate for covid and Healthy",

    xlabel="Respiration Rate"

)

# %% [markdown]

# # Q6

# %%

# Plot distributions

dataset[['T\_healty', 'T\_covid']].plot.hist(

    bins=40,

    color=[(0,1,0,0.5),(1,0,0,0.5)],

    xlabel="Temperature",

    title="Temperature Class Distribution"

)

# %%

# lable covid class

temp\_1\_df = pd.DataFrame()

temp\_1\_df["temp"] = dataset['T\_covid']

temp\_1\_df["class"] = 1

# lable healthy class

temp\_0\_df = pd.DataFrame()

temp\_0\_df["temp"] = dataset['T\_healty']

temp\_0\_df["class"] = 0

# get dataset

temp\_df = pd.concat([temp\_1\_df, temp\_0\_df])

temp\_df

# %%

# split the data for test train

X\_train, X\_test, y\_train, y\_test = train\_test\_split(temp\_df["temp"].values, temp\_df["class"].values, test\_size=0.02, random\_state=42)

X\_train = X\_train.reshape(-1, 1)

X\_test = X\_test.reshape(-1, 1)

# %%

# train model

lrc = LogisticRegression(random\_state=0).fit(X\_train, y\_train)

svm = SVC(random\_state=0).fit(X\_train, y\_train)

# %%

y\_stand = ((X\_train - X\_train.mean()) / (X\_train.std()))

y\_norm = ((X\_train - X\_train.min()) / (X\_train.max() - X\_train.min()))

# %%

y\_lrc = lrc.predict\_proba(X\_train)[:, 1]

y\_svm = svm.predict(X\_train)

fpr\_lrc, tpr\_lrc, \_ = roc\_curve(y\_train, y\_lrc)

roc\_auc\_lrc = auc(fpr\_lrc, tpr\_lrc)

fpr\_svm, tpr\_svm, \_ = roc\_curve(y\_train, y\_svm )

roc\_auc\_svm = auc(fpr\_svm, tpr\_svm)

fpr\_norm, tpr\_norm, \_ = roc\_curve(y\_train, y\_norm )

roc\_auc = auc(fpr\_norm, tpr\_norm)

plt.figure(figsize=(8, 6))

plt.plot(fpr\_lrc, tpr\_lrc, color='green', lw=2, linestyle='-.', label=f'Logistic Regression - AUC={roc\_auc\_lrc:.4f}')

plt.plot(fpr\_svm, tpr\_svm, color='orange',  lw=2, linestyle='--', label=f'SVM - AUC={roc\_auc\_svm:.4f}')

plt.plot(fpr\_norm, tpr\_norm, color='red',  lw=2, linestyle=':', label=f'Simple Classifier - AUC={roc\_auc:.4f}')

plt.plot([0, 1], [0, 1], color='black', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.0])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title(f'Receiver Operating Characteristic (ROC) Curve - AUC(Simple Classifier) = {roc\_auc:.4f}')

plt.legend(loc='lower right')

plt.show()

# %%

# References

Green, J. (2023). Part 3: Experimental design. *BrightSpace*, 20-22. Retrieved from https://brightspace.carleton.ca/d2l/le/content/212097/viewContent/3302629/View

Tutor, T. O. (2019). Chi Square Test. YouTube. Retrieved from https://www.youtube.com/watch?v=HKDqlYSLt68

Wikipedia. (n.d.). *Boxplot.* Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Interquartile\_range

Yi, M. (n.d.). A Complete Guide to Box Plots. CartIO. Retrieved from https://chartio.com/learn/charts/box-plot-complete-guide/