# Homework: Worksheet for Chapters 3:3 – 4:3

Student Name: Ben Collier

Watch: Bug fix for Book’s chapter 3:4 <https://www.screencast.com/t/JN3OSXLgr> ( 5 mins)

### Reading Assignment:

Skim Chapter 3:3 ‘Baseline feature transformation’ in the book: Reproducible Machine Learning for Credit Card Fraud Detection – Practical Handbook by Borgne et al. available free through CC BY-SA 4.0 license at [https://fraud-detection-handbook.github.io/fraud-detection-handbook/Foreword.html#](https://fraud-detection-handbook.github.io/fraud-detection-handbook/Foreword.html)

Read all of Chapter 3:4 ‘Baseline fraud detection system’ and run the code in Google Colab (45 mins).

Read Chapter 3:5 ‘Real-world data’ (5 mins)

Read Chapter 4:1 Performance Metrics Introduction (5 mins)

Read Chapter 4:2.0, 4:2.1, and 4:2.2, and skim the rest of Chapter 4:2 (20 mins)

Read Chapter 4:3 Threshold-free metrics (30 mins)

Optionally skim the rest of Chapter 4.

Answer the various questions below in a sentence or two. You may summarize in your own words or copy and paste from the book. These questions are to help draw your attention to items Dr. Humpherys considers important. The purpose is not to force you to memorize anything or have you written long-winded essays. This worksheet is to help you learn the book’s content. Do the various learning activities listed below, e.g., run code as instructed. Explore and experiment. The more you explore from the book and experiment with code, the more you will learn!

### Deliverable:

Submit this worksheet with your embedded answers to WTclass\cidm6356\lessons\Week 12\Homework: Worksheet from Chapters 3:3-4:3\

### Chapter 3:3 Baseline Feature Transformation

1. Skim Chapter 3 section 3. The main take home point is that you can transform the credit card data to give you new variables to include in your machine learning models, variables such as ‘was the transaction on a weekend or weekday’, etc. There are good fraud detection strategies in this chapter, but beyond our current scope.

### Chapter 3:4 Baseline Fraud Detection System

1. Run all the code in [Dr. Humpherys’ version of Chapter 3:4 in Colab](https://colab.research.google.com/drive/1dYgF29KiDjOpnQ7Z_E6q4fAo9In1dTF3?usp=sharing). ~~Click the Rocket icon (top right) then choose Colab.~~ The book’s version has a bug from a depreciated append() function. Dr. Humpherys corrected the bug in his version.

Focus on understanding the purpose of the code--- its input and output. You do not need to understand every line of code. Optional: If you are curious about any code, ask [Google Gemini](https://gemini.google.com/app) to explain it. You’ll be pleasantly surprised at how well Google Gemini explains code. Nothing to turn in yet.

1. After code cell [2] ‘# Load data from the 2018-07-25 to the 2018-08-14’ insert a new code cell with the following code to inspect the transformed data frame. Notice the new columns that were added to the data after Chapter 3:3. Nothing to turn in yet.

transactions\_df.head()

1. Let’s add some new ML classifiers and see how they perform at finding fraud. In about cell [21] ‘classifiers\_dictionary’, change the code from Figure 1 to Figure 2. Specifically, we are adding a support vector machine (SVM), a naïve bayes classifier called GaussianNB(), and a neural network (NN) called MLPClassifier(). It may take a minute or two to run this code.

Figure 1. Old classifiers, original code.

A close up of text

Description automatically generated

Figure 2. Change code to add additional ML Classifiers.

A close-up of a computer screen

Description automatically generated

1. Run your modified code and the next several cells. As evidence of learning, copy and paste here the results of the classifiers on the *test set* (see Chapter 3:4 ‘# performances on test set’). The output should include the results from the SVM, Naïve Bayes, and Neural Network classifiers.

|  |  |  |  |
| --- | --- | --- | --- |
| index | AUC ROC | Average precision | Card Precision@100 |
| XGBoost | 0.875 | 0.605 | 0.271 |
| SVM | 0.874 | 0.604 | 0.29 |
| Neural Network | 0.873 | 0.661 | 0.289 |
| Logistic regression | 0.871 | 0.606 | 0.291 |
| Random forest | 0.867 | 0.658 | 0.287 |
| Naive Bayes | 0.816 | 0.248 | 0.267 |

1. Which three classifiers have the highest performance according to AUC ROC and Card Precision@100? XG Boost, SVM, and Neural Network have the highest AUC ROC. As for Card Precision, Logistic Regression and SVM are the highest, and Neural Network/Random Forest are tied for the third spot. Taking the average between the two, SVM, Logistic Regression, and Neural Network are the highest.

### Chapter 3:5 Real-world data

Read this section. Nothing to turn in.

### Chapter 4:1 Performance Metrics Introduction

Read this section. Nothing to turn in.

### Chapter 4:2.0, 4:2.1, 4:2.2

1. Regarding a confusion matrix, which do you want to increase, and which do you want to minimize: TP, FP, TN, FN? For the best performance, you should aim to decrease the number of false negatives and false positives but increase the number of true positives and true negatives.
2. What is a threshold? (See Chapter 4:2.1 and 4:2.2)

Answer provided by Dr. Humpherys, “A number between 0 and 1 for which you conclude that a classification score above the threshold is considered fraudulent and scores below the threshold are legitimate. A logistic regression or other binary classification algorithm gives you a classification score between 0 and 1. You compare the logistic regression score to the threshold. Any score above the threshold number is considered fraudulent and any score below the threshold is legitimate. As a developer you choose the threshold number based on your business logic and tradeoffs (discussed later in Chapter 4:2). This threshold concepts applies to any classification problem in machine learning, not just fraud detection.”

1. What are the tradeoffs between wanting to catch all the fraud and avoiding denial of legitimate transactions?

The answer is provided for you to emphasis this point. “A fraud detection problem is in nature a cost-sensitive problem: missing a fraudulent transaction is usually considered more costly than raising a false alert on a legitimate transaction. In the former case, losses not only include the amount of the fraudulent transaction. They also include other fraudulent transactions that may be carried out using the compromised card, as well as customer services costs related to later dealing with the issue, and the reputation of the company. In the latter case, the cost is reduced by checking the legitimacy of the transaction with the customer. In case of real-time detection, it also entails the customer inconvenience related to the blocking their legitimate payment.” (Chapter 4.2.3)

### Chapter 4:3 Threshold-free metrics

The main point of this section is that the Area Under the Curve ROC (AUC ROC) is a popular metric of accuracy in machine learning. The higher the AUC ROC, the better performant the classifier. You can directly compare multiple classifiers by their AUR ROC values. However, there are limitations of AUC ROC for data sets where fraud is infrequent. The Precision-Recall curve has improvements over AUC ROC but is harder to interpret. Don’t memorize the math, focus on understanding the concept and in interpreting the metrics.

Do the following learning activity to learn how to code an AUC ROC, Precision-Recall curve, and how to use generative AI to help you learn.

1. Open a new Google Colab notebook and name it “Example AUC ROC”.
2. Click “generate with AI” in the first cell. [Note. If you do not see “generate with AI” in your notebook, you may not have access to this free feature. Instead, you can use <https://gemini.google.com/> to ask coding questions of, then copy and paste the example code into Google Colab. ]
3. Type this prompt “Please write code that does a logistical regression on sample data from sklearn and plots the AUC ROC curve.” Press Enter or click the [Generate] button. Sample code will be written in your Colab notebook.
4. Run your code.
5. Add a new cell, click “generate with AI.”
6. Type this prompt “Please write code that does a logistical regression on sample data from sklearn and plots a Precision-Recall curve.”
7. Run your code.
8. Save this Collab notebook for future reference.
9. Explore the code, particularly how to code AUC ROC and Precision-Recall plots. You’ll need this skill soon.
10. Change the Share to “Anyone with a Link” and paste the link to your Colab notebook here as evidence of learning. <https://colab.research.google.com/drive/1KfQBS5-zMKy4ePtmlo43il5Hd3yQNpPV#scrollTo=lKeN0Glhf05P>

## Grading Rubric: 100 points.

1. Answered all the questions in this worksheet. 50 points
2. Evidence of coding three additional classifiers as per instructions. 25 points
3. Evidence of coding an AUC ROC and Precision-Recall plots as per instructions. 25 points.