# LAB1: Reading Data from CSV Files

## Scenario

Greene City National Bank (GCNB) has given you access to the sources of data that you've agreed could be useful for achieving its business goals. The data is spread out among different files and databases, and is stored in different formats. From what the data owners at GCNB have told you, there are four main tables of data:

- **Users**: This table includes rows of users and various columns that describe each user's demographics and banking information. This is the largest table that GCNB provided in terms of the sheer volume of data.
- **User devices**: This table tracks what kind of device each user primarily uses when banking electronically with GCNB.
- **User transaction history**: This table includes records of individual financial transactions and the monetary value of each transaction.
- **Consumer loan complaints**: This table includes information about complaints that users have sent to GCNB when going through the loan process.

Your job is to starting pulling all of this data together so that it can eventually be cleaned and consolidated into a single working environment. The consumer loan complaint data is contained within its own comma-separated values (CSV) file, so you'll start by reading that data first.

# LAB2: Extracting Data with Database Queries

## Scenario

The remaining three tables GCNB provided to you are all contained within an SQL database. You have been given access to this database, but in order to actually retrieve the data, you'll need to execute some SQL queries. You also want to begin shaping the data so that it's in a more workable form. For example, the transaction history database currently records each transaction separately, even transactions made by the same user at different times. Rather than keep it like this, you want to aggregate the transactions for each user so that it's easier to integrate this data with the other tables when the time comes.

# LAB3: Extracting Data with Database Queries

## Scenario

You've extracted all of the relevant data from each source. The data tables currently live in separate objects, but they share a common key: `user_id`. Instead of keeping them all separate, you'll begin consolidating them using this primary key so they end up as a single table. In particular, you'll create a master table that includes user demographics, banking

information, device usage, and transaction history. Having this master table will make many forthcoming data science tasks easier.

# **LAB4:** Consolidating Data from Multiple Sources

## Scenario

You've extracted all of the relevant data from each source. The data tables currently live in separate objects, but they share a common key: **`user_id`**. Instead of keeping them all separate, you'll begin consolidating them using this primary key so they end up as a single table. In particular, you'll create a master table that includes user demographics, banking information, device usage, and transaction history. Having this master table will make many forthcoming data science tasks easier.

# **LAB5:** Handling Irregular and Unusable Data

## Scenario

The tables that GCNB sent over have thousands of records, and it's unlikely that they are in a pristine state. There's a chance they include at least some corrupt or faulty data, whether as a result of data entry errors or something else. In any case, you need to find any unusable data and deal with it so that it doesn't cause issues later on. You have a suspicion that some of the user ages may have been recorded incorrectly, so you'll look for evidence of that and take the appropriate action, if necessary. There may also be other circumstances that could indicate faulty data, so you'll do some more investigation.

# **LAB6:** Correcting Data Formats

## Scenario

Thankfully, it seems that most of the data types in the dataset are being cast correctly. However, you'll need to convert string objects to Boolean values where appropriate, since some columns should be **`True`** or **`False`**. Also, dates and times can often cause problems when they're pulled into a programming environment, especially since they're often just cast as standard strings. While this isn't necessarily a problem, it's much easier to work with dates and times when they're cast as datetime objects. So, you'll convert the relevant column from a string object to a datetime.

# LAB7: Correcting Data Formats

## Scenario

Another issue that plagues relatively large datasets is the presence of duplicates, and the GCNB data is no different. You'll identify any potentially duplicated rows and then remove them from the dataset. That way, the analysis and modeling you eventually perform won't be skewed by repeated data.

# LAB8: Handling Textual Data

## Scenario

The consumer complaints data that you worked with earlier may still be of value to the project, especially if the team plans to develop natural language processing (NLP) models sometime in the future. However, handling the textual data inside this file requires a much different approach than the numeric and categorical data that you've mostly been working with. You need a way to process the text so that it's more conducive to analysis and machine learning. There are many techniques for doing so, and you'll apply several of them to the complaint data.

# LAB9: Loading Data into a Database

## Scenario

Now that the GCNB data is in a relatively clean state, you want to begin packaging it for the forthcoming analysis and modeling tasks. There are many formats that you can load this data into; instead of choosing just one, you'll try out multiple formats to get a feel for how they differ in terms of storage and integration. First, you'll load the dataset into an SQL database.

# LAB10: Loading Data into a DataFrame

## Scenario

SQL databases are common storage platforms for relational datasets, but they aren't the only option available to you. In fact, if you plan to do a lot of work in a programming language like Python, you may want to preserve your data in a `DataFrame` structure so that it's easier to read from and write to. This is possible by saving the `DataFrame` as a binary pickle file, which you'll do in this lab.

# LAB11: Exporting Data to a CSV File

## Scenario

Some of the other team members on the project don't use Python and aren't particularly comfortable with SQL. But they still need a way to work on the same GCNB data that you've been preparing. So, you'll export the dataset as a comma-separated values (CSV) file, which preserves the data and its general structure in a text file. That way they can more easily integrate the data into whatever environments they are using.

# LAB12: Examining Data

## Scenario

Now that you've gone through the ETL process on the GCNB users data at least once, you can start taking a closer look at the characteristics of that data. There's likely still many more ways the data can be improved before you use it to start building machine learning models. You'll begin your analysis by examining the data from a high level, just to get a sense of what state it's in and what you have to work with.

# LAB13: Exploring the Underlying Distribution of Data

## Scenario

Now that you've explored the structure and format of the GCNB dataset, you can begin to dive a little deeper into the nature of the values themselves. You'll focus on exploring the distribution of these values, which should give you a sense of how they are spread out and how they can be described using various summary statistics.

# LAB14: Analyzing Data Using Histograms

## Scenario

You want to start exploring the GCNB dataset visually to aid in your interpretation. There are many chart types you can construct from this data, but you'll start with histograms in order to view the distributions of the several features.

# **LAB15:** Analyzing Data Using Box Plots and Violin Plots

## Scenario

Histograms are great for showing general distribution shape, but there are other ways to visualize the spread of values. You'll create box plots and violin plots to help you identify outliers that you may want to remove so that they don't unduly influence any models you plan on building.

# **LAB16:** Analyzing Data Using Scatter Plots and Line Plots

## Scenario

Other than using distribution plots, you can also get a sense of how data appears when two or more features are compared to each other. You want to see if the total amount of all of a user's transactions is related to the number of transactions they engage in. Perhaps the more transactions the user makes, the more of a total net positive they have in terms of total amount. Or, perhaps it's the opposite. There may even be no correlation whatsoever between the number of transactions and the total amount. So, you'll generate a scatter plot to find out.
You also want to see how account age is affecting the total amount, if at all. Are long-time users more likely to have a net positive in their account, or not? So, you'll generate a line plot comparing the years users signed up and the average total amount for each user in a sign-up year.

# **LAB17:** Analyzing Data Using Bar Charts

## Scenario

The GCNB dataset has many categorical features, which are great candidates for bar charts. You want see the frequency of values in each of these features. So, you'll create several bar charts to compare those frequencies.

# **LAB18:** Analyzing Data Using Maps

## Scenario

You've taken a preliminary look at the feature correlations in the dataset, but you want to dive a little deeper. A correlation heatmap is a good way to visually compare these feature correlations, so you'll create one. The correlations or lack thereof may dictate how you drop or keep certain features.

# **LAB19:** Handling Missing Values

## Scenario

As you've probably seen by now, the GCNB dataset has a lot of missing data. You need to handle that missing data before it can be used to build a machine learning model. Rather than remove all of the missing data wholesale, you'll take a more surgical approach and apply different methods to different instances.

# **LAB20:** Applying Transformation Functions to a Dataset

## Scenario

You've seen that a lot of the numeric features in the dataset are skewed. Adjusting this skewness can better facilitate the model training process. These types of adjustments can be dependent on the type of model you're training, so for now, you'll just temporarily transform the `age` feature to get a sense of how such transformations can help mitigate distribution issues. Later, when you start building machine learning models, you'll apply other transformations to the dataset.

# **LAB21:** Encoding Data

## Scenario

Since you plan to input this data into multiple machine learning algorithms, you need to make sure the data is actually in a form that those algorithms can read. Many machine learning algorithms can't deal with categorical features that use string values, so those values need to be encoded as numbers. For the most part, you'll use the common one-hot encoding technique to ensure that the features are properly formatted for machine learning.

# LAB22: Encoding Data

## Scenario

When you eventually build machine learning models using certain types of algorithms, you'll need to transform your continuous variables into discrete ones. You'll start with `age`, which you'll place into one of several bins. These bins will act as categories to a new feature, which you'll use to replace the original continuous `age` variable.

# LAB23: Splitting and Removing Features

## Scenario

Although the `date_joined` feature is useful, machine learning algorithms don't always work well with complex dates and times. You'll simplify the data by extracting just the months from this feature, and then make that its own feature. This could be useful for identifying things like churn rate, e.g., perhaps there's a higher churn rate in summer months than in winter.
You also want to see what remaining features you can potentially remove to improve the dataset's viability. Earlier you started identifying correlations between pairs of variables, so now you'll actually start dropping one variable in each of those high-correlation pairs. This will hopefully remove redundancy in the dataset. Likewise, you want to identify any features that exhibit low variance and drop them. Features with very low variance have values that are so similar they may not contain any truly useful information.
Lastly, with all of the preprocessing done on this dataset, you'll finally be ready to load it into a data file that you'll use with machine learning.

# LAB24: Performing Dimensionality Reduction

## Scenario

Your users dataset now has dozens of features, which will likely be useful in many machine learning scenarios. However, some machine learning tasks are more effective when they work with a reduced feature set. Likewise, there are times when you'll want to reduce your dataset's dimensionality to make it more conducive to plotting. There are only so many dimensions you can represent on a scatter plot, for instance.
So, you'll apply principal component analysis (PCA) to a subset of your data—demographics information, in particular—to reduce its dimensionality. Because most of your work will be done on the full users dataset, you'll save this as a separate dataset that you can call up whenever you need it.

# **LAB25:** Training Logistic Regression Models

## Scenario

The time has come to start building machine learning models that will help GCNB make intelligent decisions about its data and its business operations. One of the potential target variables you and your team identified earlier was the `term_deposit` variable. The more effective a GCNB marketing campaign, the more term deposits the bank can get from its customers, and the more revenue it can generate. Your task is therefore to determine what customers are most likely to sign up for a term deposit, so that the marketing campaigns can prioritize those customers and seek out new customers who have the features that are most important for predicting a term deposit. This is a good task for classification.
Rather than just build one classification model and call it a day, you'll end up building multiple models from various algorithms to determine which one best meets your performance needs. You'll start by training a logistic regression model. Before you train the model, you'll need to do a bit more prep work on the data to make it more conducive to machine learning.
**Note**: By default, these lab steps assume you will be typing the code shown in screenshots. Many learners find it easier to understand what code does when they are able to type it themselves. However, if you prefer not to type all of code, the **Solutions** folders contain the finished code for each notebook.

# **LAB26:** Training a k-NN Model

## Scenario

The initial logistic regression model didn't produce very good results. But, there are plenty more classification algorithms to try. So, you'll build another model, this time using *k*-nearest neighbor (*k*-NN), a distanced-based algorithm. You'll make predictions using this model and evaluate its accuracy score as well to see how it compares.

# **LAB27:** Training an SVM Classification Model

## Scenario

The *k*-NN model did better than the logistic regression model, but there's still a chance to improve your scores and generate a better model. So, you'll try out your data on a support-vector machine (SVM) algorithm. SVMs excel at modeling data with outliers. Although you dealt with outliers earlier in the ETL process, an SVM model might still perform well. So, you'll train one and find out.

# LAB28: Training a Naïve Bayes Model

## Scenario

The next model you'll train is a naïve Bayes model, which calculates classification probabilities based on Bayes' theorem. These types of models assume that the features do not interact with each other, which they almost certainly do in a dataset like this one. Still, it's worth trying out, so you'll build one and see how it scores.

# LAB29: Training Classification Decision Trees and Ensemble Models

## Scenario

As you've seen, individually trained models can produce worthwhile results, but you may also get better results with ensemble methods. Ensemble algorithms use an aggregation of multiple decision trees to classify data. So, you'll leverage these algorithms by first training a single decision tree, then training multiple trees within random forests and gradient boosting models. Hopefully, these methods will improve upon the individual models you've built thus far.

# LAB30: Tuning Classification Models

## Scenario

You've built several different models to classify the `term_deposit` target variable. Each model is different in how it makes its predictions, as well as its success at doing so. However, effective machine learning employs an iterative tuning process, where such models can get better when they're configured properly. You'll go through this tuning process by determining optimal hyperparameters for two different models: the logistic regression model and the gradient boosting model. The former obviously performed very poorly at first, so it should only get better through tuning. The latter has the highest score thus far, but may be susceptible to overfitting (as tree-based models often are). So, you'll see if you can improve that model as well, while also reducing overfitting through cross-validation.

# LAB31: Evaluating Classification Models

## Scenario

Now that you've trained and tuned your models, you wanted to evaluate their performance using more than just accuracy. There are many useful classification metrics that might be relevant to your data. In particular, because you balanced the dataset through oversampling, and because neither recall nor precision are more valuable in the case of predicting users

signing up for a term deposit, you decide to focus on $F_1$ score as the primary metric. Still, it's important to consider other metrics as well, so you'll get a more comprehensive look at your models.

# LAB32: Training a Linear Regression Model

## Scenario

Now that you're done developing classification models, you want to turn your attention to developing models that can estimate continuous numeric values. The `total_amount_usd` that you created in the ETL phase by consolidating transactional data is of particular interest. It represents the total balance of each user's transactions with the bank over the given period of time. So, the `total_amount_usd` reveals either the debit that the user owes to the bank (negative values), or the credit that the bank extends to the user (positive values). You want to build models that can estimate a new customer's debit/credit based on various features.
As with your classification tasks, you won't just build one model, but several. You'll compare these regressors to determine which one best meets your needs. To start with, you'll train a very simple linear model to see how it handles the data.

# LAB33: Training Regression Trees and Ensemble Models

## Scenario

The simpler linear regression model was able to make predictions on the `total_amount_usd` target variable, but there's definitely room for improvement. Decision trees and ensemble models are not just useful for classification, but regression as well. So you'll train several of these models to see if you can get some better results.

# LAB34: Tuning Regression Models

## Scenario

You've trained some basic regression models, and now you can begin to tune them. You want to tune the linear regression model to see if you can improve its performance. Having a well-performing simple linear model would be beneficial since it's easier to explain and isn't as prone to overfitting as some others. You also want to see if you can reduce the high levels of overfitting in your lone decision tree. Again, having a well-performing simple model has its advantages.
You could also try to improve the XGBoost model, but in the interest of time, you'll just focus on the linear model and the decision tree for now.

# LAB35: Evaluating Regression Models

## Scenario

Your preliminary training would suggest that some of your regression models are better than others. But, there are more ways to evaluate these models, and you want to be sure you have a more comprehensive understanding of their performance. So, you'll compare each model using multiple metrics, then take a deeper look at one of the models you think is best.

# LAB36: Training a k-Means Clustering Model

## Scenario

Although supervised learning has been very useful to the project so far, there are other ways you can learn more from GCNB's customer data that don't involve labels. The marketing department might shape their campaigns to maximize term deposit subscriptions, but that's just one of many factors that define customers and their behavior. The business would benefit greatly from being able to target specific segments of customers for their marketing campaigns. Targeted marketing can lead to greater customer engagement with many different facets of the business. After all, the bank offers more services than just term deposits.
So, you decide to see how you can segment the bank's customers using cluster analysis. There are a few different methods you can try, but you'll start by building a model that leverages the *k*-means algorithm for generating clusters.

# LAB37: Training a Hierarchical Clustering Model

## Scenario

The *k*-means approach is the most common method for cluster analysis in unsupervised learning, and it seems to be a good choice for the GCNB dataset. Hierarchical clustering is less common and has more limited applications, and may not be as suitable for this dataset. Still, like any other type of machine learning, it's always a good idea to try out multiple algorithms when time permits. So, you'll generate a hierarchical clustering model to see how it groups customers.

# LAB38: Tuning Clustering Models

## Scenario

You developed an initial *k*-means clustering model based on an arbitrary choice of 5 clusters. However, you want to adjust the model to see how it changes in response to a different number of clusters. Also, earlier in the preprocessing phase, you reduced the dimensionality of the user demographics data. So, you'll load this reduced dataset to make the clusters easier to visualize on a chart.

# LAB39: Evaluating Clustering Models

## Scenario

As with supervised learning, you'll need some way to evaluate your clustering models. This is especially important for choosing the optimal number of clusters, as any change to this number will have a significant impact on how GCNB's customers are segmented.

# LAB40: Building an ML Pipeline

## Scenario

The GCNB marketing project is coming to a close, but that doesn't mean your work is finished. It was good for you to go through the entire data science process, but your approach wasn't necessarily the most efficient. It would help you save time and effort if you were able to automate the process, making it repeatable for future projects. What you need is to set up a data pipeline. Eventually, the team wants to look into cloud-based options for setting up a pipeline that can automate the entire data science process—from data collection all the way to presentation.

For now, however, you want to test out scikit-learn's `Pipeline` module, which focuses on automating the machine learning model training process. The tasks you performed earlier to train, tune, and evaluate models can be condensed and streamlined using this pipeline functionality. This pipeline will be able to take any new set of training data and build a working model from it. However, because the pipeline only focuses on the model training process, it will assume that any new training data fed into it will already have been cleaned. In this case, you'll implement the pipeline using the familiar GCNB customer dataset, with classification as the goal. After you create the pipeline, you'll test the results on a new set of unlabeled customer data so that you can predict whether or not these new customers will sign up for a term deposit.

**Note**: By default, these lab steps assume you will be typing the code shown in screenshots. Many learners find it easier to understand what code does when they are able to type it themselves. However, if you prefer not to type all of code, the **Solutions** folders contain the finished code for each notebook.