Quantitative Methods

Big Data Projects

by Sreekanth Mallikarjun, PhD, and Ahmed Abbasi, PhD

Sreekanth Mallikarjun, PhD, is at Reorg (USA) and the University of Virginia, McIntire School of Commerce (USA). Ahmed Abbasi, PhD, is at the University of Virginia, McIntire School of Commerce (USA).

LEARNING OUTCOMES

Mastery	*The candidate should be able to:*
□	**a.** state and explain steps in a data analysis project;
□	**b.** describe objectives, steps, and examples of preparing and wrangling data;
□	**c.** describe objectives, methods, and examples of data exploration;
□	**d.** describe objectives, steps, and techniques in model training;
□	**e.** describe preparing, wrangling, and exploring text-based data for financial forecasting;
□	**f.** describe methods for extracting, selecting and engineering features from textual data;
□	**g.** evaluate the fit of a machine learning algorithm.

INTRODUCTION

1

Big data (also referred to as alternative data) encompasses data generated by financial markets (e.g., stock and bond prices), businesses (e.g., company financials, production volumes), governments (e.g., economic and trade data), individuals (e.g., credit card purchases, social media posts), sensors (e.g., satellite imagery, traffic patterns), and the Internet of Things, or IoT, (i.e., the network of interrelated digital devices that can transfer data among themselves without human interaction). A veritable explosion in big data has occurred over the past decade or so, especially in unstructured data generated from social media (e.g., posts, tweets, blogs), email and text communications, web traffic, online news sites, electronic images, and other electronic information sources. The prospects are for exponential growth in big data to continue.

Investment managers are increasingly using big data in their investment processes as they strive to discover signals embedded in such data that can provide them with an information edge. They seek to augment structured data with a plethora of unstructured data to develop improved forecasts of trends in asset prices, detect anomalies, etc. A typical example involves a fund manager using financial text data from 10-K reports for forecasting stock sentiment (i.e., positive or negative), which can then be used as an input to a more comprehensive forecasting model that includes corporate financial data.

Unlike structured data (numbers and values) that can be readily organized into data tables to be read and analyzed by computers, unstructured data typically require specific methods of preparation and refinement before being usable by machines (i.e., computers) and useful to investment professionals. Given the volume, variety, and velocity of available big data, it is important for portfolio managers and investment analysts to have a basic understanding of how unstructured data can be transformed into structured data suitable as inputs to machine learning (ML) methods (in fact, for any type of modeling methods) that can potentially improve their financial forecasts.

This reading describes the steps in using big data, both structured and unstructured, in financial forecasting. The concepts and methods are then demonstrated in a case study of an actual big data project. The project uses text-based data derived from financial documents to train an ML model to classify text into positive or negative sentiment classes for the respective stocks and then to predict sentiment.

Section 2 of the reading covers a description of the key characteristics of big data. Section 3 provides an overview of the steps in executing a financial forecasting project using big data. We then describe in Sections 4–6 key aspects of data preparation and wrangling, data exploration, and model training using structured data and unstructured (textual) data. In Section 7, we bring these pieces together by covering the execution of an actual big data project. A summary in Section 8 concludes the reading.

2 BIG DATA IN INVESTMENT MANAGEMENT

Big data differs from traditional data sources based on the presence of a set of characteristics commonly referred to as the 3Vs: volume, variety, and velocity.

Volume refers to the quantity of data. The US Library of Congress, which is tasked with archiving both digital and physical information artifacts in the United States, has collected hundreds of terabytes of data (one terabyte equals 1,024 gigabytes, which are equal to 1,048,576 megabytes). Several years ago, one of the authors managed an archival project for the Library of Congress in which many terabytes of online content were collected—a copious amount of data at the time. However, in most US industry sectors today, the average company collects more data than the Library of Congress! In big data conversations, terabytes have been replaced with petabytes and exabytes (one exabyte equals 1,024 petabytes, which are equal to 1,048,576 terabytes). The classic grains of sand analogy puts these volumes into perspective: If a megabyte is a tablespoon of sand, then a petabyte is a 1.6-kilometer-long beach and an exabyte is a beach extending about 1,600 kilometers.

Variety pertains to the array of available data sources. Organizations are now dealing with structured, semi-structured, and unstructured data from within and outside the enterprise. Variety includes traditional transactional data; user-generated text, images, and videos; social media; sensor-based data; web and mobile clickstreams; and spatial-temporal data. Effectively leveraging the variety of available data presents both opportunities and challenges, including such legal and ethical issues as data privacy.

Velocity is the speed at which data are created. Many large organizations collect several petabytes of data every hour. With respect to unstructured data, more than one billion new tweets (i.e., a message of 280 characters or less posted on the social media website Twitter) are generated every three days; five billion search queries occur daily. Such information has important implications for real-time predictive analytics in various financial applications. Analyzing such "data-in-motion" poses challenges since relevant patterns and insights might be moving targets relative to situations of "data-at-rest."

When using big data for inference or prediction, there is a "fourth V": *Veracity relates to the credibility and reliability of different data sources.* Determining the credibility and reliability of data sources is an important part of any empirical investigation. The issue of veracity becomes critically important for big data, however, because of the varied sources of these large datasets. Big data amplifies the age-old challenge of disentangling quality from quantity. Social media, including blogs, forums, and social networking sites, are plagued with spam; by some estimates, as much as 10%–15% of such content is completely fake. Similarly, according to our research, web spam accounts for more than 20% of all content on the worldwide web. Clickstreams from website and mobile traffic are equally susceptible to noise. Furthermore, deriving deep semantic knowledge from text remains challenging in certain instances despite significant advances in natural language processing (NLP).

These Vs have numerous implications for financial technology (commonly referred to as "fintech") pertaining to investment management. Machine learning assessments of creditworthiness, which have traditionally relied on structured financial metrics, are being enhanced by incorporating text derived from financial statements, news articles, and call transcripts. Customers in the financial industry are being segmented based not only on their transactional data but also on their views and preferences expressed on social media (to the degree permissible under applicable privacy agreements). Big data also affords opportunities for enhanced fraud detection and risk management.

STEPS IN EXECUTING A DATA ANALYSIS PROJECT: FINANCIAL FORECASTING WITH BIG DATA

3

In the era of big data, firms treat data like they do important assets. However, effective big data analytics are critical to allow appropriate data monetization. Let us take financial forecasting as an application area. Numerous forecasting tasks in this domain can benefit from predictive analytics models built using machine learning methods. One common example is predicting whether stock prices (for an individual stock or a portfolio) will go up or down in value at some specific point in the future. Traditionally, financial forecasting relied on various financial and accounting numbers, ratios, and metrics coupled with statistical or mathematical models. More recently, machine learning models have been commonly utilized. However, with the proliferation of textual big data (e.g., online news articles, internet financial forums, social networking platforms), such unstructured data have been shown to offer insights faster (as they are real-time) and have enhanced predictive power.

Textual big data provides several valuable types of information, including topics and sentiment. Topics are what people are talking about (e.g., a firm, an industry, a particular event). Sentiment is how people feel about what they are discussing. For instance, they might express positive, negative, or neutral views (i.e., sentiments) toward a topic of discussion. One study conducted in the United States found that positive sentiment on Twitter could predict the trend for the Dow Jones Industrial Average up to three days later with nearly 87% accuracy.

Deriving such insights requires supplementing traditional data with textual big data. As depicted in Exhibit 1, the inclusion of big data has immediate implications for building the machine learning model as well as downstream implications for financial forecasting and analysis. We begin with the top half of Exhibit 1, which shows the traditional (i.e., with structured data) *ML Model Building Steps:*

1 *Conceptualization of the modeling task.* This crucial first step entails determining what the output of the model should be (e.g., whether the price of a stock will go up/down one week from now), how this model will be used and by whom, and how it will be embedded in existing or new business processes.

2 *Data collection.* The data traditionally used for financial forecasting tasks are mostly numeric data derived from internal and external sources. Such data are typically already in a structured tabular format, with columns of features, rows of instances, and each cell representing a particular value.

3 *Data preparation and wrangling.* This step involves cleansing and preprocessing of the raw data. Cleansing may entail resolving missing values, out-of-range values, and the like. Preprocessing may involve extracting, aggregating, filtering, and selecting relevant data columns.

4 *Data exploration.* This step encompasses exploratory data analysis, feature selection, and feature engineering.

5 *Model training.* This step involves selecting the appropriate ML method (or methods), evaluating performance of the trained model, and tuning the model accordingly.

Note that these steps are iterative because model building is an iterative process. The insights gained from one iteration may inform the next iteration, beginning with reconceptualization. In contrast with structured data sources, textual big data originating in online news articles, social media, internal/external documents (such as public financial statements), and other openly available data sources are unstructured.

The *Text ML Model Building Steps* used for the unstructured data sources of big data are shown in the bottom half of Exhibit 1. They differ from those used for traditional data sources and are typically intended to create output information that is structured. The differences in steps between the text model and traditional model account for the characteristics of big data: volume, velocity, variety, and veracity. In this reading, we mostly focus on the variety and veracity dimensions of big data as they manifest themselves in text. The major differences in the *Text ML Model Building Steps* are in the first four steps:

1 *Text problem formulation.* Analysts begin by determining how to formulate the text classification problem, identifying the exact inputs and outputs for the model. Perhaps we are interested in computing sentiment scores (structured output) from text (unstructured input). Analysts must also decide how the text ML model's classification output will be utilized.

2 *Data (text) curation.* This step involves gathering relevant external text data via web services or **web spidering (scraping or crawling) programs** that extract raw content from a source, typically web pages. Annotation of the text data with high-quality, reliable target (dependent) variable labels might also be necessary for supervised learning and performance evaluation purposes. For instance, experts might need to label whether a given expert assessment of a stock is bearish or bullish.

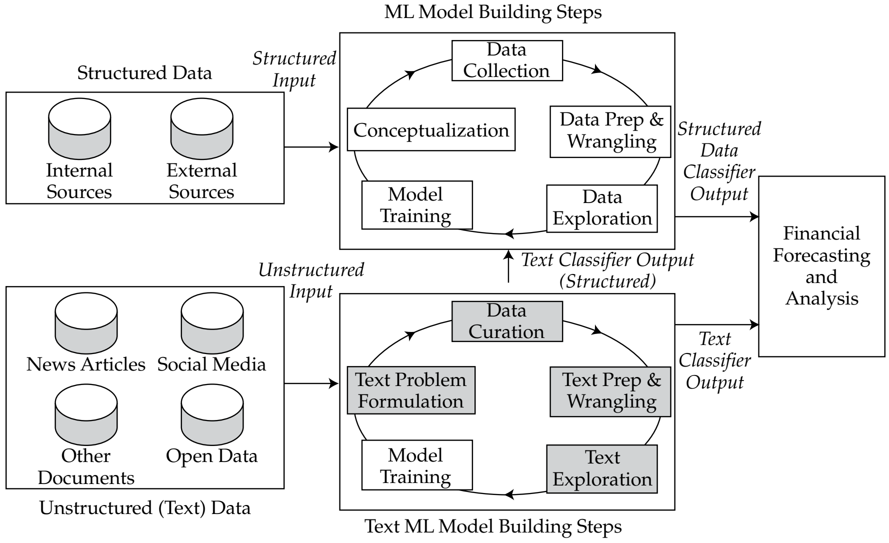
3 *Text preparation and wrangling.* This step involves critical cleansing and preprocessing tasks necessary to convert streams of unstructured data into a format that is usable by traditional modeling methods designed for structured inputs.

4 *Text exploration.* This step encompasses text visualization through techniques, such as word clouds, and text feature selection and engineering.

The resulting output (e.g., sentiment prediction scores) can either be combined with other structured variables or used directly for forecasting and/or analysis.

Next, we describe two key steps from the *ML Model Building Steps* depicted in Exhibit 1 that typically differ for structured data versus textual big data: data/text preparation and wrangling and data/text exploration. We then discuss model training. Finally, we focus on applying these steps to a case study related to classifying and predicting stock sentiment from financial texts.

Exhibit 1 Model Building for Financial Forecasting Using Big Data: Structured (Traditional) vs. Unstructured (Text)



EXAMPLE 1

Steps in ML Model Building

LendALot Corporation is a B2C (business-to-consumer) lender that has traditionally outsourced potential customers' creditworthiness scoring to a third-party firm. Given the recent advances in machine learning (ML)-based "fintech" that goes beyond traditional "repayment history" and "ability to repay" assessments derived from structured data, LendALot would like to develop in-house, ML-based credit scoring capabilities to enhance borrower risk assessment and differentiate itself in the B2C lending market. LendALot would like to follow a phased approach beginning with traditional (structured) data sources and then eventually incorporating textual (unstructured) big data sources. Paul Wang has been asked to lead a new analytics team at LendALot tasked with developing the ML-based creditworthiness scoring model. In the context of machine learning using structured data sources, address the following questions.

1 State and explain one decision Wang will need to make related to:

A conceptualizing the modeling task.

B data collection.

C data preparation and wrangling.

D data exploration.

E model training.

In a later phase of the project, LendALot attempts to improve its credit scoring processes by incorporating textual data in credit scoring. Wang tells his team, "Enhance the creditworthiness scoring model by incorporating insights from text provided by the prospective borrowers in the loan application free response fields."

2 Identify the process step that Wang's statement addresses.

3 State two potential needs of the LendAlot team in relation to text curation.

4 State two potential needs of the LendAlot team in relation to text preparation and wrangling.

Solution to 1:

A In the conceptualization step, Wang will need to decide how the output of the ML model will be specified (e.g., a binary classification of creditworthiness), how the model will be used and by whom, and how it will be embedded in LendALot's business processes.

B In the data collection phase, Wang must decide on what data—internal, external, or both—to use for credit scoring.

C In the data preparation and wrangling step, Wang will need to decide on data cleansing and preprocessing needs. Cleansing may entail resolving missing values, extreme values, etc. Preprocessing may involve extracting, aggregating, filtering, and selecting relevant data columns.

D In the data exploration phase, Wang will need to decide which exploratory data analysis methods are appropriate, which features to use in building a credit scoring model, and which features may need to be engineered.

E In the model training step, Wang must decide which ML algorithm(s) to use. Assuming labeled training data are available, the choice will be among supervised learning algorithms. Decisions will need to be made on how model fit is measured and how the model is validated and tuned.

Solution to 2:

Wang's statement relates to the initial step of text problem formulation.

Solution to 3:

Related to text curation, the team will be using internal data (from loan applications). They will need to ensure that the text comment fields on the loan applications have been correctly implemented and enabled. If these fields are not required, they need to ensure there is a sufficient response rate to analyze.

Solution to 4:

Related to text preparation and wrangling, the team will need to carry out the critical tasks of text cleansing and text preprocessing. These two tasks are necessary to convert an unstructured stream of data into structured values for use by traditional modeling methods.

DATA PREPARATION AND WRANGLING

4

Data preparation and wrangling involve cleansing and organizing raw data into a consolidated format. The resulting dataset is suitable to use for further analyses and training a machine learning (ML) model. This is a critical stage, the foundation, in big data projects. Most of the project time is spent on this step, and the quality of the data affects the training of the selected ML model. Domain knowledge—that is, the involvement of specialists in the particular field in which the data are obtained and used—is beneficial and often necessary to successfully execute this step. Data preparation is preceded by data collection, so we discuss the data collection process first.

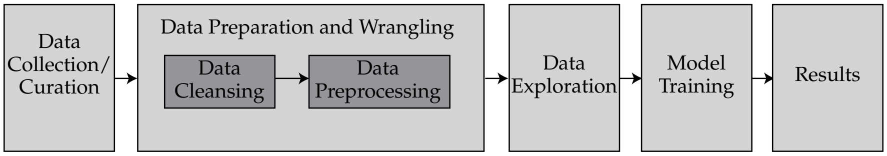
Before the data collection process even begins, it is important to state the problem, define objectives, identify useful data points, and conceptualize the model. Conceptualization is like a blueprint on a drawing board, a modifiable plan that is necessary to initiate the model building process. A project overview is established by determining the ML model type—supervised or unsupervised—and data sources/collection plans with respect to the needs of the project.

Data collection involves searching for and downloading the raw data from one or multiple sources. Data can be stored in different formats, sources, and locations. As databases are the most common primary sources, building necessary queries with the help of database administrators is critical. Database schemas are built with certain assumptions and exceptions, and it is safest to clarify the database architecture with an administrator or database architect before downloading the necessary data. Data also exist in the form of spreadsheets, comma-separated values (csv) files, text files, and other formats. Care must be taken before using such data, and documentation (often referred to as "Readme" files) must be referred to, if available. **Readme files** are text files provided with the raw data that contain information related to a data file. They are useful for understanding the data and how they can be interpreted correctly.

Alternatively, third-party data vendors can be sources of clean data. External data usually can be accessed through an **application programming interface (API)**—a set of well-defined methods of communication between various software components—or the vendors can deliver the required data in the form of csv files or other formats (as previously mentioned). Using external data can save time and resources that would otherwise go into data preparation and wrangling; however, vendor contracts come with a price. Depending on the big data project constraints, a decision must be made regarding the use of internal or external data based on the trade-offs between time, financial costs, and accuracy. For projects using internal user data, external data might not be suitable. For example, to understand user traffic on a company website, internally recorded site visits and click frequency may be captured and stored in the internal databases. External data are advantageous when a project requires generic data, such as demographics of a geographic area or traffic data of a public service. Another consideration in using external vendor provided data is that during the cleansing process, underlying trends in the data that are important for particular end-uses may be masked or even lost. This is where "alpha" is often found; so by simply buying a dataset from a vendor, you may lose your information edge. Of course, application of the data (e.g., merging and combining, putting through different types of models) will be different for everyone who uses it; there are always different ways to extract value.

Once the data are collected, the data preparation and wrangling stage begins. This stage involves two important tasks: cleansing and preprocessing, respectively. Exhibit 2 outlines data preparation and wrangling and defines the two component tasks. These tasks are explained in detail under the structured and unstructured subsections because the steps vary by the nature of data.

Exhibit 2 Data Preparation and Wrangling Stage



Data Preparation (Cleansing): This is the initial and most common task in data preparation that is performed on raw data. Data cleansing is the process of examining, identifying, and mitigating errors in raw data. Normally, the raw data are neither sufficiently complete nor sufficiently clean to directly train the ML model. Manually entered data can have incomplete, duplicated, erroneous, or inaccurate values. Automated data (recorded by systems) can have similar problems due to server failures and software bugs.

Data Wrangling (Preprocessing): This task performs transformations and critical processing steps on the cleansed data to make the data ready for ML model training. Raw data most commonly are not present in the appropriate format for model consumption. After the cleansing step, data need to be processed by dealing with outliers, extracting useful variables from existing data points, and scaling the data.

4.1 Structured Data

Data Preparation (Cleansing)

Structured data are organized in a systematic format that is readily searchable and readable by computer operations for processing and analyzing. In structured data, data errors can be in the form of incomplete, invalid, inaccurate, inconsistent, non-uniform, and duplicate data observations. The data cleansing process mainly deals with identifying and mitigating all such errors. Exhibit 3 shows a raw dataset before cleansing. The data have been collected from different sources and are organized in a data matrix (or data table) format. Each row contains observations of each customer of a US-based bank. Each column represents a variable (or feature) corresponding to each customer.

Exhibit 3 Raw Data Before Cleansing

1	ID	Name	Gender	Date of Birth	Salary	Other Income	State	Credit Card
2	1	Mr. ABC	M	12/5/1970	$50,200	$5,000	VA	Y
3	2	Ms. XYZ	M	15 Jan, 1975	$60,500	$0	NY	Yes
4	3	EFG		1/13/1979	$65,000	$1,000	CA	No
5	4	Ms. MNO	F	1/1/1900	—	—	FL	Don't Know

Exhibit 3 (Continued)

1	ID	Name	Gender	Date of Birth	Salary	Other Income	State	Credit Card
6	5	Ms. XYZ	F	15/1/1975	$60,500	$0		Y
7	6	Mr. GHI	M	9/10/1942	NA	$55,000	TX	N
8	7	Mr. TUV	M	2/27/1956	$300,000	$50,000	CT	Y
9	8	Ms. DEF	F	4/4/1980	$55,000	$0	British Columbia	N

The possible errors in a raw dataset include the following:

1 *Incompleteness error* is where the data are not present, resulting in missing data. This can be corrected by investigating alternate data sources. Missing values and NAs (not applicable or not available values) must be either omitted or replaced with "NA" for deletion or substitution with imputed values during the data exploration stage. The most common imputations are mean, median, or mode of the variable or simply assuming zero. In Exhibit 3, rows 4 (ID 3), 5 (ID 4), 6 (ID 5), and 7 (ID 6) are incomplete due to missing values in either Gender, Salary, Other Income, Name (Salutation), and State columns.

2 *Invalidity error* is where the data are outside of a meaningful range, resulting in invalid data. This can be corrected by verifying other administrative data records. In Exhibit 3, row 5 likely contains invalid data as the date of birth is out of the range of the expected human life span.

3 *Inaccuracy error* is where the data are not a measure of true value. This can be rectified with the help of business records and administrators. In Exhibit 3, row 5 is inaccurate (it shows "Don't Know"); in reality, every person either has a credit card or does not.

4 *Inconsistency error* is where the data conflict with the corresponding data points or reality. This contradiction should be eliminated by clarifying with another source. In Exhibit 3, row 3 (ID 2) is likely to be inconsistent as the Name column contains a female title and the Gender column contains male.

5 *Non-uniformity error* is where the data are not present in an identical format. This can be resolved by converting the data points into a preferable standard format. In Exhibit 3, the data under the Date of Birth column is present in various formats. The data under the Salary column may also be non-uniform as the monetary units are ambiguous; the dollar symbol can represent US dollar, Canadian dollar, or others.

6 *Duplication error* is where duplicate observations are present. This can be corrected by removing the duplicate entries. In Exhibit 3, row 6 is a duplicate as the data under Name and Date of Birth columns are identical to the ones in row 3, referring to the same customer.

Exhibit 4 shows the dataset after completion of the cleansing process.

Exhibit 4. Data After Cleansing

1	ID	Name	Gender	Date of Birth	Salary	Other Income	State	Credit Card
2	1	Mr. ABC	M	12/5/1970	USD 50200	USD 5000	VA	Y
3	2	Ms. XYZ	F	1/15/1975	USD 60500	USD 0	NY	Y
4	3	Mr. EFG	M	1/13/1979	USD 65000	USD 1000	CA	N
5	6	Mr. GHI	M	9/10/1942	USD 0	USD 55000	TX	N
6	7	Mr. TUV	M	2/27/1956	USD 300000	USD 50000	CT	Y
7	8	Ms. DEF	F	4/4/1980	CAD 55000	CAD 0	British Columbia	N

Data cleansing can be expensive and cumbersome because it involves the use of automated, rule-based, and pattern recognition tools coupled with manual human inspection to sequentially check for the aforementioned types of errors row by row and column by column. The process involves a detailed data analysis as an initial step in identifying various errors that are present in the data. In addition to a manual inspection and verification of the data, analysis software, such as SPSS, can be used to understand **metadata** (data that describes and gives information about other data) about the data properties to use as a starting point to investigate any errors in the data. The business value of the project determines the necessary quality of data cleansing and subsequently the amount of resources used in the cleansing process. In case the errors cannot be resolved due to lack of available resources, the data points with errors can simply be omitted depending on the size of the dataset. For instance, if a dataset is large with more than 10,000 rows, removing a few rows (approximately 100) may not have a significant impact on the project. If a dataset is small with less than 1,000 rows, every row might be important and deleting many rows thus harmful to the project.

Data Wrangling (Preprocessing)

To make structured data ready for analyses, the data should be preprocessed. Data preprocessing primarily includes transformations and scaling of the data. These processes are exercised on the cleansed dataset. The following transformations are common in practice:

1. *Extraction:* A new variable can be extracted from the current variable for ease of analyzing and using for training the ML model. In Exhibit 4, the Date of Birth column consists of dates that are not directly suitable for analyses. Thus, an additional variable called "Age" can be extracted by calculating the number of years between the present day and date of birth.
2. *Aggregation:* Two or more variables can be aggregated into one variable to consolidate similar variables. In Exhibit 4, the two forms of income, Salary and Other Income, can be summed into a single variable called Total Income.
3. *Filtration:* The data rows that are not needed for the project must be identified and filtered. In Exhibit 4, row 7 (ID 8) has a non-US state; however, this dataset is for the US-based bank customers where it is required to have a US address.

4 *Selection:* The data columns that are intuitively not needed for the project can be removed. This should not be confused with feature selection, which is explained later. In Exhibit 4, Name and Date of Birth columns are not required for training the ML model. The ID column is sufficient to identify the observations, and the new extracted variable Age replaces the Date of Birth column.

5 *Conversion:* The variables can be of different types: nominal, ordinal, continuous, and categorical. The variables in the dataset must be converted into appropriate types to further process and analyze them correctly. This is critical for ML model training. Before converting, values must be stripped out with prefixes and suffixes, such as currency symbols. In Exhibit 4, Name is nominal, Salary and Income are continuous, Gender and Credit Card are categorical with 2 classes, and State is ordinal. In case row 7 is not excluded, the Salary in row 7 must be converted into US dollars. Also, the conversion task applies to adjusting time value of money, time zones, and others when present.

Outliers may be present in the data, and domain knowledge is needed to deal with them. Any outliers that are present must first be identified. The outliers then should be examined and a decision made to either remove or replace them with values imputed using statistical techniques. In Exhibit 4, row 6 (ID 7) is an outlier because the Salary value is far above the upper quartile. Row 5 (ID 6) is also an outlier because the Salary value is far below the lower quartile. However, after the aggregation and formation of a new variable Total Income, as shown in Exhibit 5, row 5 (ID 6), it is no longer an outlier.

In practice, several techniques can be used to detect outliers in the data. Standard deviation can be used to identify outliers in normally distributed data. In general, a data value that is outside of 3 standard deviations from the mean may be considered an outlier. The interquartile range (IQR) can be used to identify outliers in data with any form of distribution. IQR is the difference between the 75th and the 25th percentile values of the data. The center of the IQR is the median (50th percentile). In general, data values outside of 1.5 IQR are considered as outliers and values outside of 3 IQR as extreme values.

There are several practical methods for handling outliers. When extreme values and outliers are simply removed from the dataset, it is known as **trimming** (also called truncation). For example, a 5% trimmed dataset is one for which the 5% highest and the 5% lowest values have been removed. When extreme values and outliers are replaced with the maximum (for large value outliers) and minimum (for small value outliers) values of data points that are not outliers, the process is known as **winsorization**.

Exhibit 5 Data After Applying Transformations

1	ID	Gender	Age	Total Income	State	Credit Card
2	1	M	48	55200	VA	Y
3	2	F	43	60500	NY	Y
4	3	M	39	66000	CA	N
5	6	M	76	55000	TX	N

Scaling is a process of adjusting the range of a feature by shifting and changing the scale of data. Variables, such as age and income, can have a diversity of ranges that result in a heterogeneous training dataset. For better ML model training when using such methods as support vector machines (SVMs) and artificial neural networks

(ANNs), all variables should have values in the same range to make the dataset homogeneous. It is important to remove outliers before scaling is performed. Here are two of the most common ways of scaling:

1 *Normalization* is the process of rescaling numeric variables in the range of [0, 1]. To normalize variable X, the minimum value (X_{min}) is subtracted from each observation (X_i), and then this value is divided by the difference between the maximum and minimum values of X ($X_{\text{max}} - X_{\text{min}}$) as follows:

$$X_{i\ \text{(normalized)}} = \frac{X_i - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{1}$$

2 *Standardization* is the process of both centering and scaling the variables. Centering involves subtracting the mean (μ) of the variable from each observation (X_i) so the new mean is 0. Scaling adjusts the range of the data by dividing the centered values ($X_i - \mu$) by the standard deviation (σ) of feature X. The resultant standardized variable will have an arithmetic mean of 0 and standard deviation of 1.

$$X_{i\ \text{(standardized)}} = \frac{X_i - \mu}{\sigma} \tag{2}$$

Normalization is sensitive to outliers, so treatment of outliers is necessary before normalization is performed. Normalization can be used when the distribution of the data is not known. Standardization is relatively less sensitive to outliers as it depends on the mean and standard deviation of the data. However, the data must be normally distributed to use standardization.

EXAMPLE 2

Preparing and Wrangling Structured Data

Paul Wang's analytics team at LendALot Corporation is working to develop its first ML model for classifying prospective borrowers' creditworthiness. Wang has asked one of his data scientists, Lynn Lee, to perform a preliminary assessment of the data cleansing and preprocessing tasks the team will need to perform. As part of this assessment, Lee pulled the following sample of data for manual examination, which she brings to Wang to discuss.

1	ID	Name	Loan Outcome	Income (USD)	Loan Amount (USD)	Credit Score	Loan Type
2	1	Mr. Alpha	No Default	34,000	10,000	685	Mortgage
3	2	Ms. Beta	No Default	–63,050	49,000	770	Student Loan
4	3	Mr. Gamma	Defaulted	20,565	35,000	730	
5	4	Ms. Delta	No Default	50,021	unknown	664	Mortgage
6	5	Mr. Epsilon	Defaulted	100,350	129,000	705	Car Loan
7	6	Mr. Zeta	No Default	800,000	300,000	800	Boat Loan
8	6	Mr. Zeta	No Default	800,000	300,000	800	Boat Loan

After sharing a concern that the data should be thoroughly cleansed, Wang makes the following statements:

Statement 1 "Let's keep the ID column and remove the column for Name from the dataset."

Statement 2 “Let’s create a new feature, “Loan Amount as a Percent of Income,” to use as an additional feature.”

1 The data shown for Ms. Beta contain what is *best described* as an:

A invalidity error.

B inaccuracy error.

C incompleteness error.

2 The data shown for Mr. Gamma contain what is *best described* as an:

A invalidity error.

B duplication error.

C incompleteness error.

3 The data shown for Ms. Delta contain what is *best described* as an:

A invalidity error.

B inaccuracy error.

C duplication error.

4 The data shown for Mr. Zeta contain what is *best described* as an:

A invalidity error.

B inaccuracy error.

C duplication error.

5 The process mentioned in Wang’s first statement is *best described* as:

A feature selection.

B feature extraction.

C feature engineering

6 Wang’s second statement is *best described* as:

A feature selection.

B feature extraction.

C feature engineering.

Solution to 1:

A is correct. This is an invalidity error because the data are outside of a meaningful range. Income cannot be negative.

Solution to 2:

C is correct. This is an incompleteness error as the loan type is missing.

Solution to 3:

B is correct. This is an inaccuracy error because LendALot must know how much they have lent to that particular borrower (who eventually repaid the loan as indicated by the loan outcome of no default).

Solution to 4:

C is correct. Row 8 duplicates row 7: This is a duplication error.

Solution to 5:

A is correct. The process mentioned involves selecting the features to use. The proposal makes sense; with “ID,” “Name” is not needed to identify an observation.

Solution to 6:

B is correct. The proposed feature is a ratio of two existing features. *Feature extraction* is the process of creating (i.e., extracting) new variables from existing ones in the data.

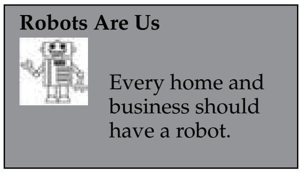
4.2 Unstructured (Text) Data

Unstructured data are not organized into any systematic format that can be processed by computers directly. They are available in formats meant for human usage rather than computer processing. Unstructured data constitute approximately 80% of the total data available today. They can be in the form of text, images, videos, and audio files. Unlike in structured data, preparing and wrangling unstructured data are both more challenging. For analysis and use to train the ML model, the unstructured data must be transformed into structured data. In this section, text data will be used to demonstrate unstructured data preparation and wrangling. The cleansing and preprocessing of text data is called *text processing*. Text processing is essentially cleansing and transforming the unstructured text data into a structured format. Text processing can be divided into two tasks: cleansing and preprocessing. The following content is related to text data in the English language.

Text Preparation (Cleansing)

Raw text data are a sequence of characters and contain other non-useful elements, including html tags, punctuations, and white spaces (including tabs, line breaks, and new lines). It is important to clean the text data before preprocessing. Exhibit 6 shows a sample text from the home page for the hypothetical company Robots Are Us website. The text appears to be clean visually and is designed for human readability.

Exhibit 6 Sample Text from Robots Are Us Home Page



Robots Are Us

Every home and business should have a robot.

However, the source text that can be downloaded is not as clean. The raw text contains html tags and formatting elements along with the actual text. Exhibit 7 shows the raw text from the source.

Exhibit 7 Raw Text from the Source

```
<h1 class="text-left mb-3">Robots Are Us</h1>
<h2> Every home and business shoudl have a robot    </h2>
```

The initial step in text processing is cleansing, which involves basic operations to clean the text by removing unnecessary elements from the raw text. Text operations often use regular expressions. A **regular expression (regex)** is a series that contains characters in a particular order. Regex is used to search for patterns of interest in a given text. For example, a regex "<.*?>" can be used to find all the html tags that are present in the form of <…> in text.[1] GREP (global regular expression print) is a commonly available utility in programming languages for searching patterns using regex. Once a pattern is found, it can be removed or replaced. Additionally, advanced html parsers and packages are available in the popular programming languages, such as R and Python, to deal with this task.

The following steps describe the basic operations in the text cleansing process.

1 *Remove html tags*: Most of the text data are acquired from web pages, and the text inherits html markup tags with the actual content. The initial task is to remove (or strip) the html tags that are not part of the actual text using programming functions or using regular expressions. In Exhibit 7, </h2> is an html tag that can be identified by a regex and be removed. Note that it is not uncommon to keep some generic html tags to maintain certain formatting meaning in the text.

2 *Remove Punctuations*: Most punctuations are not necessary for text analysis and should be removed. However, some punctuations, such as percentage signs, currency symbols, and question marks, may be useful for ML model training. These punctuations should be substituted with such annotations as /percentSign/, /dollarSign/, and /questionMark/ to preserve their grammatical meaning in the text. Such annotations preserve the semantic meaning of important characters in the text for further text processing and analysis stages. It is important to note that periods (dots) in the text need to be processed carefully. There are different circumstances for periods to be present in text—characteristically used for abbreviations, sentence boundaries, and decimal points. The periods and the context in which they are used need to be identified and must be appropriately replaced or removed. In general, periods after abbreviations can be removed, but the periods separating sentences should be replaced by the annotation /endSentence/. Some punctuations, such as hyphens and underscores, can be kept in the text to keep the consecutive words intact as a single term (e.g., e-mail). Regex are often used to remove or replace punctuations.

3 *Remove Numbers*: When numbers (or digits) are present in the text, they should be removed or substituted with an annotation /number/. This helps inform the computer that a number is present, but the actual value of the number itself is not helpful for categorizing/analyzing the text. Such operations are critical for ML model training. Otherwise, the computers will treat each number as a separate word, which may complicate the analyses or add noise. Regex are often used to remove or replace numbers. However, the number and any decimals must be retained where the outputs of interest are the actual values of the number. One such text application is information extraction (IE), where the goal is to extract relevant information from a given text. An IE task could be extracting monetary values from financial reports, where the actual number values are critical.

4 *Remove white spaces*: It is possible to have extra white spaces, tab spaces, and leading and ending spaces in the text. The extra white spaces may be introduced after executing the previously mentioned operations. These should be identified and removed to keep the text intact and clean. Certain functions in programming languages can be used to remove unnecessary white spaces from the text. For example, the text mining package in R offers a *stripwhitespace* function.

[1] A regex of the form "<.*?>" will identify all html tags with anything (*) of any length (?) between the brackets (< >).

Exhibit 8 uses a sample financial text to show the transformations occurring after applying each operation of the text cleansing process. The four steps are applied on a mock financial text after scraping from a source. As noted previously, scraping (or web scraping) is a technique to extract raw content from a source, typically web pages. It is important to note that the sequence and choice of cleansing operations does matter. For instance, after removing punctuation, the "1.2 million" becomes "12 million." This is acceptable here since a subsequent operation replaces all numbers with a "/number/" tag. However, if numbers were not replaced with such tags, the punctuation removal operation could affect the data.

Exhibit 8 Text Cleansing Process Example

Original text from a financial statement as shown on a webpage

CapEx on the normal operations remained stable on historicallylow levels, $800,000 compared to $1.2 million last year.

Quarter 3, so far, is 5% sales growth quarter-to-date, and year-to-date, we have a 4% local currency sales development.

Raw text after scraping from the source

<p><font size = "4"> CapEx on the normal operations remained stable on historically low levels, $800,000 compared to $1.2 million last year. <b/><b/> Quarter 3, so far, is 5% sales growth quarter-to-date, and year-to-date, we have a 4% local currency sales development.</font></p>

Text after removing html tags

CapEx on the normal operations remained stable on historically low levels, $800,000 compared to $1.2 million last year.
Quarter 3, so far, is 5% sales growth quarter-to-date, and year-to-date, we have a 4% local currency sales development.

Text after removing and replacing punctuations

CapEx on the normal operations remained stable on historically low levels /dollarSign/800000 compared to /dollarSign/12 million last year /endSentence/ Quarter 3 so far is 5 /percentSign/ sales growth quarter-to-date and year-to-date we have a 4 /percentSign/ local currency sales development /endSentence/

Text after replacing numbers

CapEx on the normal operations remained stable on historically low levels /dollarSign//number / compared to/dollarSign//number/ million last year /endSentence/ Quarter/number/ so far is /number/ /percentSign/sales growth quarter-to-date and year-to-date we have a /number/ / percentSign/ local currency sales development /endSentence/

Text after removing extra white spaces

CapEx on the normal operations remained stable on historically low levels/dollarSign//number /compared to/dollarSign//number/million last year/endSentence/ Quarter/number/so far is /number//percentSign/sales growth quarter-to-date and year-to-date we have a/number// percentSign/local currency sales development/endSentence/

Text Wrangling (Preprocessing)

To further understand text processing, tokens and tokenization need to be defined. A **token** is equivalent to a word, and **tokenization** is the process of splitting a given text into separate tokens. In other words, a text is considered to be a collection of tokens. Tokenization can be performed at word or character level, but it is most commonly performed at word level. Exhibit 9 shows a sample dataset of four cleansed texts and their word tokens.

Exhibit 9 Tokenization of Four Texts

	Cleaned Texts	Tokens						
Text 1	The man went to the market today	The	man	went	to	the	market	today
Text 2	Market values are increasing	Market	values	are	increasing			
Text 3	Increased marketing is needed	Increased	marketing	is	needed			
Text 4	There is no market for the product	There	is	no	market	for	the	product

Similar to structured data, text data also require normalization. The normalization process in text processing involves the following:

1 *Lowercasing* the alphabet removes distinctions among the same words due to upper and lower cases. This action helps the computers to process the same words appropriately (e.g., “The” and “the”).

2 *Stop words* are such commonly used words as “the,” “is,” and “a.” Stop words do not carry a semantic meaning for the purpose of text analyses and ML training. However, depending on the end-use of text processing, for advance text applications it may be critical to keep the stop words in the text in order to understand the context of adjacent words. For ML training purposes, stop words typically are removed to reduce the number of tokens involved in the training set. A predefined list of stop words is available in programming languages to help with this task. In some cases, additional stop words can be added to the list based on the content. For example, the word “exhibit” may occur often in financial filings, which in general is not a stop word but in the context of the filings can be treated as a stop word.

3 *Stemming* is the process of converting inflected forms of a word into its base word (known as stem). Stemming is a rule-based approach, and the results need not necessarily be linguistically sensible. Stems may not be the same as the morphological root of the word. Porter’s algorithm is the most popular method for stemming. For example, the stem of the words “analyzed” and “analyzing” is “analyz.” Similarly, the British English variant “analysing” would become “analys.” Stemming is available in R and Python. The text mining package in R provides a *stemDocument* function that uses this algorithm.

4 *Lemmatization* is the process of converting inflected forms of a word into its morphological root (known as lemma). Lemmatization is an algorithmic approach and depends on the knowledge of the word and language structure. For example, the lemma of the words “analyzed” and “analyzing” is “analyze.” Lemmatization is computationally more expensive and advanced.

Stemming or lemmatization will reduce the repetition of words occurring in various forms and maintain the semantic structure of the text data. Stemming is more common than lemmatization in the English language since it is simpler to perform. In text data, data sparseness refers to words that appear very infrequently, resulting in data consisting of many unique, low frequency tokens. Both techniques decrease data sparseness by aggregating many sparsely occurring words in relatively less sparse stems or lemmas, thereby aiding in training less complex ML models.

After the cleansed text is normalized, a bag-of-words is created. **Bag-of-words (BOW)** representation is a basic procedure used to analyze text. It is essentially a collection of a distinct set of tokens from all the texts in a sample dataset. BOW is simply a set of words and does not capture the position or sequence of words present in the text. However, it is memory efficient and easy to handle for text analyses.

Exhibit 10 shows the BOW and transformations occurring in each step of normalization on the cleansed texts from Exhibit 9. Note that the number of words decreases as the normalizing steps are applied, making the resulting BOW smaller and simpler.

Exhibit 10 Bag-of-Words Representation of Four Texts Before and After Normalization Process

BOW before normalizing

“The”	“man”	“went”	“to”	“the”	“market”
“today”	“Market”	“values”	“are”	“increasing”	“Increased”
“marketing”	“is”	“needed”	“There”	“no”	“for”
“product”					

BOW after removing uppercase letters

“the”	“man”	“went”	“to”	“market”	“today”
“values”	“are”	“increasing”	“increased”	“marketing”	“is”
“needed”	“there”	“no”	“for”	“product”	

BOW after removing stop words

“man”	“went”	“market”	“today”	“values”	“increasing”
“increased”	“marketing”	“needed”	“product”		

BOW after stemming

“man”	“went”	“market”	“today”	“valu”	“increas”	“need”	“product”

The last step of text preprocessing is using the final BOW after normalizing to build a **document term matrix (DTM)**. DTM is a matrix that is similar to a data table for structured data and is widely used for text data. Each row of the matrix belongs to a document (or text file), and each column represents a token (or term). The number of rows of DTM is equal to the number of documents (or text files) in a sample dataset. The number of columns is equal to the number of tokens from the BOW that is built using all the documents in a sample dataset. The cells can contain the counts of the number of times a token is present in each document. The matrix cells can be filled with other values that will be explained in the financial forecasting project section of this reading; a large dataset is helpful in understanding the concepts. At this point, the unstructured text data are converted to structured data that can be processed further and used to train the ML model. Exhibit 11 shows a DTM constructed from the resultant BOW of the four texts from Exhibit 10.

Exhibit 11 DTM of Four Texts and Using Normalized BOW Filled with Counts of Occurrence

	man	went	market	today	valu	increas	need	product
Text 1	1	1	1	1	0	0	0	0
Text 2	0	0	1	0	1	1	0	0
Text 3	0	0	1	0	0	1	1	0
Text 4	0	0	1	0	0	0	0	1

As seen in Exhibit 10, BOW does not represent the word sequences or positions, which limits its use for some advanced ML training applications. In the example, the word “no” is treated as a single token and has been removed during the normalization because it is a stop word. Consequently, this fails to signify the negative meaning (“no market”) of the text (i.e., Text 4). To overcome such problems, a technique called n-grams can be employed. **N-grams** is a representation of word sequences. The length of a sequence can vary from 1 to *n*. When one word is used, it is a unigram; a two-word sequence is a bigram; and a 3-word sequence is a trigram; and so on. Exhibit 10, for example, shows a unigram (*n* = 1) BOW. The advantage of n-grams is that they can be used in the same way as unigrams to build a BOW. In practice, different n-grams can be combined to form a BOW and eventually be used to build a DTM. Exhibit 12 shows unigrams, bigrams, and trigrams. Exhibit 12 also shows a combined unigram-to-trigram BOW for the particular text. Stemming can be applied on the cleansed text before building n-grams and BOW (not shown in Exhibit 12).

Exhibit 12 N-Grams and N-Grams BOW

Clean text

The man went to the market today

Unigrams

“The” “man” “went” “to” “the” “market” “today”

Bigrams

“The_man” “man_went” “went_to” “to_the” “the_market” “market_today”

Trigrams

“The_man_went” “man_went_to” “went_to_the” “to_the_market” “the_market_today”

BOW before normalizing

“The”	“man”	“went”	“to”	“the”	“market”	“today”
“The_man”	“man_went”	“went_to”	“to_the”	“the_market”	“market_today”	“The_man_went”
“man_went_to”	“went_to_the”	“to_the_market”	“the_market_today”			

BOW after removing upper case letters

“the”	“man”	“went”	“to”	“market”	“today”	“the_man”
“man_went”	“went_to”	“to_the”	“the_market”	“market_today”	“the_man_went”	“man_went_to”
“went_to_the”	“to_the_market”	“the_market_today”				

BOW after removing stop words

“man”	“went”	“market”	“today”	“the_man”	“man_went”	“went_to”
“to_the”	“the_market”	“market_today”	“the_man_went”	“man_went_to”	“went_to_the”	“to_the_market”
“the_market_today”						

The n-grams implementation will vary the impact of normalization on the BOW. Even after removing isolated stop words, stop words tend to persist when they are attached to their adjacent words. For instance, “to_the” (Exhibit 12) is a single bigram token consisting of stop words and will not be removed by the predetermined list of stop words.

EXAMPLE 3

Unstructured Data Preparation and Wrangling

1 The output produced by preparing and wrangling textual data is best described as a:

A data table.

B confusion matrix.

C document term matrix.

2 In text cleansing, situations in which one may need to add an annotation include the removal of:

A html tags.

B white spaces.

C punctuations.

3 A column of a document term matrix is *best* described as representing:

A a token.

B a regularization term.

C an instance.

4 A cell of a document term matrix is *best* described as containing:

A a token.

B a count of tokens.

C a count of instances.

5 Points to cover in normalizing textual data include:

A removing numbers.

B removing white spaces.

C lowercasing the alphabet.

6 When some words appear very infrequently in a textual dataset, techniques that may address the risk of training highly complex models include:

A stemming.

B scaling.

C data cleansing.

7 Which of the following statements concerning tokenization is *most* accurate?

A Tokenization is part of the text cleansing process.

B Tokenization is most commonly performed at the character level.

C Tokenization is the process of splitting a given text into separate tokens.

Solution to 1:

C is correct. The objective of data preparation and wrangling of textual data is to transform the unstructured data into structured data. The output of these processes is a document term matrix that can be read by computers. The document term matrix is similar to a data table for structured data.

Solution to 2:

C is correct. Some punctuations, such as percentage signs, currency symbols, and question marks, may be useful for ML model training, so when such punctuations are removed annotations should be added.

Solution to 3:

A is correct. Each column of a document term matrix represents a token from the bag-of-words that is built using all the documents in a sample dataset.

Solution to 4:

B is correct. A cell in a document term matrix contains a count of the number of tokens of the kind indicated in the column heading.

Solution to 5:

C is correct. The other choices are related to text cleansing.

Solution to 6:

A is correct. Stemming, the process of converting inflected word forms into a base word (or stem), is one technique that can address the problem described.

Solution to 7:

C is correct, by definition. The other choices are not true.

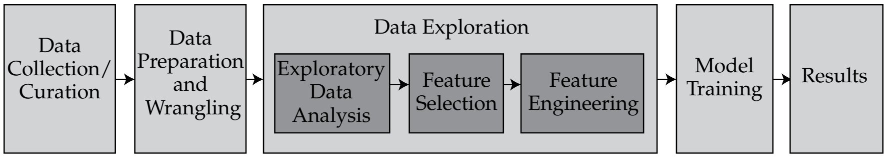
DATA EXPLORATION OBJECTIVES AND METHODS

5

Data exploration is a crucial part of big data projects. The prepared data are explored to investigate and comprehend data distributions and relationships. The knowledge that is gained about the data in this stage is used throughout the project. The outcome and quality of exploration strongly affects ML model training results. Domain knowledge plays a vital role in exploratory analysis as this stage should involve cooperation between analysts, model designers, and experts in the particular data domain. Data exploration without domain knowledge can result in ascertaining spurious relationships among the variables in the data that can mislead the analyses. The data exploration stage follows the data preparation stage and leads to the model training stage.

Data exploration involves three important tasks: exploratory data analysis, feature selection, and feature engineering. These three tasks are outlined in Exhibit 13 and are defined and further explained under the structured and unstructured data subsections.

Exhibit 13 Data Exploration Stage


Data Collection/ Curation
Data Preparation and Wrangling
Data Exploration
Exploratory Data Analysis
Feature Selection
Feature Engineering
Model Training
Results

Exploratory data analysis (EDA) is the preliminary step in data exploration. Exploratory graphs, charts, and other visualizations, such as heat maps and word clouds, are designed to summarize and observe data. In practice, many exploratory graphs are made for investigation and can be made swiftly using statistical programming and generic spreadsheet software tools. Data can also be summarized and examined using quantitative methods, such as descriptive statistics and central tendency measures. An important objective of EDA is to serve as a communication medium among project stakeholders, including business users, domain experts, and analysts. Relatively quick and easy exploratory visualizations help stakeholders connect and ensure the prepared data are sensible. Other objectives of EDA include:

- understanding data properties,
- finding patterns and relationships in data,
- inspecting basic questions and hypotheses,
- documenting data distributions and other characteristics, and
- planning modeling strategies for the next steps.

Feature selection is a process whereby only pertinent features from the dataset are selected for ML model training. Selecting fewer features decreases ML model complexity and training time. **Feature engineering** is a process of creating new features by changing or transforming existing features. Model performance heavily depends on feature selection and engineering.

5.1 Structured Data

Exploratory Data Analysis

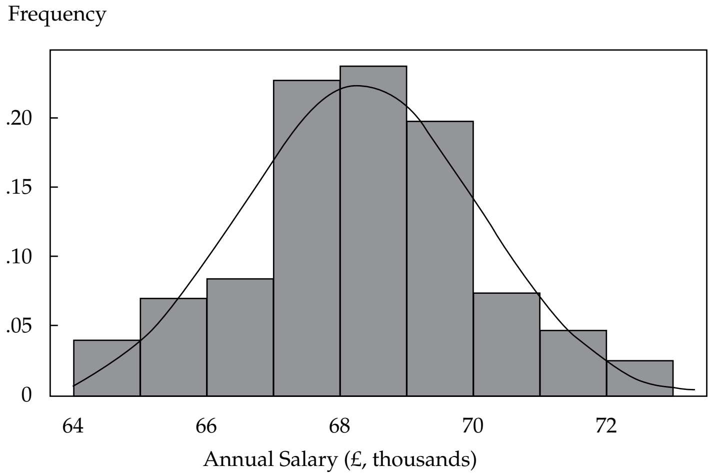
For structured data, each data table row contains an observation and each column contains a feature. EDA can be performed on a single feature (one-dimension) or on multiple features (multi-dimension). For high-dimension data with many features, EDA can be facilitated by using a dimension reduction technique, such as principal components analysis (PCA). Based on the number of dimensions, the exploratory techniques will vary.

For one-dimensional data, summary statistics, such as mean, median, quartiles, ranges, standard deviations, skewness, and kurtosis, of a feature can be computed. One-dimension visualization summarizes each feature in the dataset. The basic one-dimension exploratory visualizations are as follows:

- Histograms
- Bar charts
- Box plots
- Density plots

Histograms represent equal bins of data and their respective frequencies. They can be used to understand the high-level distribution of the data. Bar charts summarize the frequencies of categorical variables. Box plots show the distribution of continuous data by highlighting the median, quartiles, and outliers of a feature that is normally distributed. Density plots are another effective way to understand the distribution of continuous data. Density plots are smoothed histograms and are commonly laid on top of histograms, as shown in Exhibit 14. This histogram shows a hypothetical annual salary distribution (in £) of entry-level analyst positions at UK banks. The data represent a normal distribution with an approximate mean of £68,500.

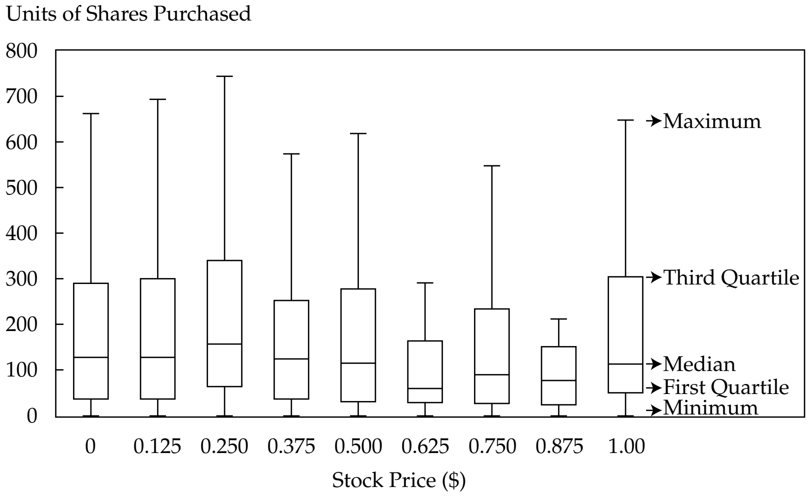
Exhibit 14 Histogram with Superimposed Density Plot



For data with two or more dimensions, summary statistics of relationships, such as a correlation matrix, can be calculated. Two- or more-dimensional visualization explores interactions between different features in the dataset. Common methods include scatterplots and line graphs. In multi-dimensional visualization, one-dimensional plots are overlaid to summarize each feature, thus enabling comparison between features. Additionally, attributes (e.g., color, shape, and size) and legends can be used creatively to pack more information about the data into fewer graphs.

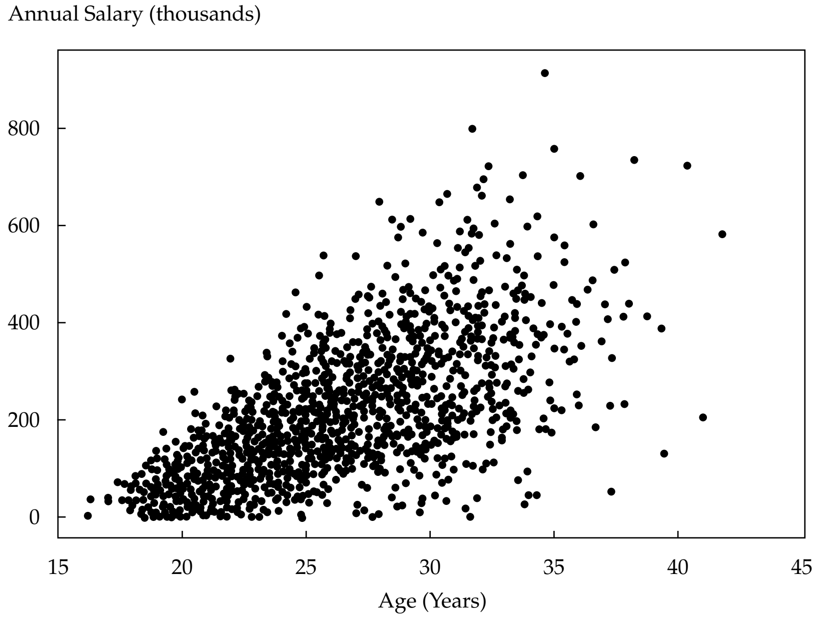
For multivariate data, commonly utilized exploratory visualization designs include stacked bar and line charts, multiple box plots, and scatterplots showing multivariate data that use different colors or shapes for each feature. Multiple box plots can be arranged in a single chart, where each individual box plot represents a feature. Such a multi-box plot chart assesses the relationship between each feature (x-axis) in the dataset and the target variable of interest (y-axis). The multi-box plot chart in Exhibit 15 represents units of shares purchased versus stock price for a hypothetical stock. The x-axis shows the stock price in increments of $0.125, and the y-axis shows units of shares purchased. The individual box plots indicate the distribution of shares purchased at the different stock prices. When the stock price is $0.25, the median number of shares purchased is the highest; when the stock price is $0.625, the median number of shares purchased is the lowest. However, visually it appears that the number of shares purchased at different stock prices is not significantly different.

Exhibit 15 Multiple Box Plots in One Chart



Two-dimensional charts can summarize and approximately measure relationships between two or more features. An example scatterplot in Exhibit 16 shows the interaction of two hypothetical features: age (x-axis) and annual salary (y-axis). The feature on the y-axis tends to increase as the feature on the x-axis increases. This pattern appears true visually; however, it may not be a statistically significant relationship. A scatterplot provides a starting point where relationships can be examined visually. These potential relationships should be tested further using statistical tests. Common parametric statistical tests include ANOVA, *t*-test, and Pearson correlation. Common non-parametric statistical tests include chi-square and the Spearman rank-order correlation.

Exhibit 16 Scatterplot Showing a Linear Relationship Between Two Features



In addition to visualization, descriptive statistics are a good means to summarize data. Central tendency measures as well as minimum and maximum values for continuous data are useful. Counts and frequencies for categorical data are commonly employed to gain insight regarding the distribution of possible values.

EDA is not only useful for revealing possible relationships among features or general trends in the data; it is also beneficial during the feature selection and engineering stages. These possible relationships and trends in the data may be used to suggest new features that, when incorporated into a model, may improve model training.

Feature Selection

Structured data consist of features, represented by different columns of data in a table or matrix. After using EDA to discover relevant patterns in the data, it is essential to identify and remove unneeded, irrelevant, and redundant features. Basic diagnostic testing should also be performed on features to identify redundancy, heteroscedasticity, and multi-collinearity. The objective of the feature selection process is to assist in identifying significant features that when used in a model retain the important patterns and complexities of the larger dataset while requiring fewer data overall. This last point is important since computing power is not free (i.e., explicit costs and processing time).

Typically, structured data even after the data preparation step can contain features that do not contribute to the accuracy of an ML model or that negatively affect the quality of ML training. The most desirable outcome is a parsimonious model with fewer features that provides the maximum predictive power out-of-sample.

Feature selection must not be confused with the data preprocessing steps during data preparation. Good feature selection requires an understanding of the data and statistics, and comprehensive EDA must be performed to assist with this step. Data preprocessing needs clarification only from data administrators and basic intuition (e.g., salary vs. income) during data preparation.

Feature selection on structured data is a methodical and iterative process. Statistical measures can be used to assign a score gauging the importance of each feature. The features can then be ranked using this score and either retained or eliminated from the dataset. The statistical methods utilized for this task are usually univariate and consider each feature independently or with regard to the target variable. Methods include chi-square test, correlation coefficients, and information-gain measures (i.e., *R*-squared values from regression analysis). All of these statistical methods can be combined in a manner that uses each method individually on each feature, automatically performing backward and forward passes over features to improve feature selection. Prebuilt feature selection functions are available in popular programming languages used to build and train ML models.

Dimensionality reduction assists in identifying the features in the data that account for the greatest variance between observations and allows for the processing of a reduced volume of data. Dimensionality reduction may be implemented to reduce a large number of features, which helps reduce the memory needed and speed up learning algorithms. Feature selection is different from dimensionality reduction, but both methods seek to reduce the number of features in the dataset. The dimensionality reduction method creates new combinations of features that are uncorrelated, whereas feature selection includes and excludes features present in the data without altering them.

Feature Engineering

After the appropriate features are selected, feature engineering helps further optimize and improve the features. The success of ML model training depends on how well the data are presented to the model. The feature engineering process attempts to produce good features that describe the structures inherent in the dataset. This process depends on the context of the project, domain of the data, and nature of the problem. Structured data are likely to contain quantities, which can be engineered to better present relevant patterns in the dataset. This action involves engineering an existing feature into a new feature or decomposing it into multiple features.

For continuous data, a new feature may be created—for example, by taking the logarithm of the product of two or more features. As another example, when considering a salary or income feature, it may be important to recognize that different salary brackets impose a different taxation rate. Domain knowledge can be used to decompose an income feature into different tax brackets, resulting in a new feature: "income_above_100k," with possible values 0 and 1. The value 1 under the new feature captures the fact that a subject has an annual salary of more than $100,000. By grouping subjects into income categories, assumptions about income tax can be made and utilized in a model that uses the income tax implications of higher and lower salaries to make financial predictions.

For categorical data, for example, a new feature can be a combination (e.g., sum or product) of two features or a decomposition of one feature into many. If a single categorical feature represents education level with five possible values—high school, associates, bachelor's, master's, and doctorate—then these values can be decomposed into five new features, one for each possible value (e.g., is_highSchool, is_doctorate) filled with 0s (for false) and 1s (for true). The process in which categorical variables are converted into binary form (0 or 1) for machine reading is called **one hot encoding**. It is one of the most common methods for handling categorical features in text data. When date-time is present in the data, such features as "second of the hour," "hour of the day," and "day of the date" can be engineered to capture critical information about temporal data attributes—which are important, for example, in modeling trading algorithms.

Feature engineering techniques systemically alter, decompose, or combine existing features to produce more meaningful features. More meaningful features allow an ML model to train more swiftly and easily. Different feature engineering strategies can lead to the generation of dramatically different results from the same ML model. The impact of feature selection and engineering on ML training is discussed further in the next section.

5.2 Unstructured Data: Text Exploration

Exploratory Data Analysis

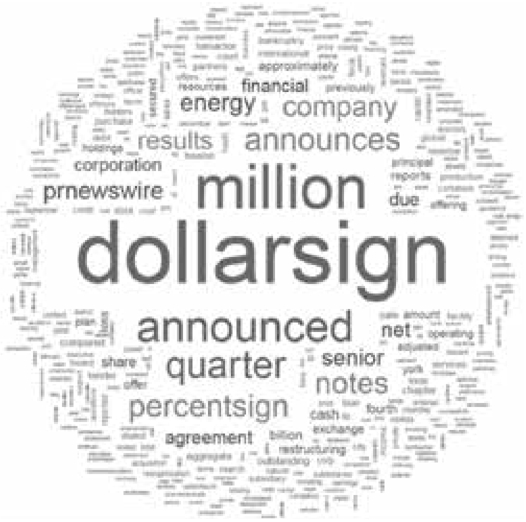
Just like with structured data, it is important to gain insight into existing patterns in the unstructured data for further analysis. In this section, text data will be discussed. Text analytics has various applications. The most common applications are text classification, topic modeling, fraud detection, and sentiment analysis. Text classification uses supervised ML approaches to classify texts into different classes. Topic modeling uses unsupervised ML approaches to group the texts in the dataset into topic clusters. Sentiment analysis predicts sentiment (negative, neutral, or positive) of the texts in a dataset using both supervised and unsupervised approaches.

Various statistics are used to explore, summarize, and analyze text data. Text data include a collection of texts (also known as a corpus) that are sequences of tokens. It is useful to perform EDA of text data by computing on the tokens such basic text statistics as **term frequency (TF)**, the ratio of the number of times a given token occurs in all the texts in the dataset to the total number of tokens in the dataset (e.g., word associations, average word and sentence length, and word and syllable counts).

Text statistics reveal patterns in the co-occurrence of words. There are many applications of text analytics, and necessary text statistics vary according to the context of the application. Topic modeling is a text data application in which the words that are most informative are identified by calculating the TF of each word. For example, the word "soccer" can be informative for the topic "sports." The words with high TF values are eliminated as they are likely to be stop words or other common vocabulary words, making the resulting BOW compact and more likely to be relevant to topics within the texts. In sentiment analysis and text classification applications, the chi-square measure of word association can be useful for understanding the significant word appearances in negative and positive sentences in the text or in different documents. The chi-square measure is further explained under feature selection. Such EDA plays a vital role in executing the feature selection step.

Text statistics can be visually comprehended by using the same methods as explained in the structured data section. For example, bar charts can be used to show word counts or frequency. Words clouds are common visualizations when working with text data as they can be made to visualize the most informative words and their TF values. The most commonly occurring words in the dataset can be shown by varying font size, and color is used to add more dimensions, such as frequency and length of words. Exhibit 17 shows a word cloud constructed from a sample dataset of generic financial news wires after text processing. Word cloud building functions and packages are available in several popular programming languages. A detailed demonstration of text data EDA will be presented in Section 7, where we work with actual text data in a financial forecasting project.

Exhibit 17 Word Cloud of Generic Financial Newsfeed Data Sample



Feature Selection

For text data, feature selection involves selecting a subset of the terms or tokens occurring in the dataset. The tokens serve as features for ML model training. Feature selection in text data effectively decreases the size of the vocabulary or BOW. This helps the ML model be more efficient and less complex. Another benefit is to eliminate noisy features from the dataset. Noisy features are tokens that do not contribute to ML model training and actually might detract from the ML model accuracy.

Noisy features are both the most frequent and most sparse (or rare) tokens in the dataset. On one end, noisy features can be stop words that are typically present frequently in all the texts across the dataset. On the other end, noisy features can be sparse terms that are present in only a few text cases. Text classification involves dividing text documents into assigned classes (a class is a category; examples include "relevant" and "irrelevant" text documents or "bearish" and "bullish" sentences). The *frequent* tokens strain the ML model to choose a decision boundary among the texts as the terms are present across all the texts, an example of model *underfitting*. The *rare* tokens mislead the ML model into classifying texts containing the rare terms into a specific class, an example of model *overfitting*. Identifying and removing noise features is very critical for text classification applications. The general feature selection methods in text data are as follows:

1 *Frequency* measures can be used for vocabulary pruning to remove noise features by filtering the tokens with very high and low TF values across all the texts. **Document frequency (DF)** is another frequency measure that helps to discard the noise features that carry no specific information about the text class and are present across all texts. The DF of a token is defined as the number of documents (texts) that contain the respective token divided by the total number of documents. It is the simplest feature selection method and often performs well when many thousands of tokens are present.

2 *Chi-square* test can be useful for feature selection in text data. The chi-square test is applied to test the independence of two events: occurrence of the token and occurrence of the class. The test ranks the tokens by their usefulness to each class in text classification problems. Tokens with the highest chi-square test statistic values occur more frequently in texts associated with a particular class and therefore can be selected for use as features for ML model training due to higher discriminatory potential.

3 *Mutual information* (MI) measures how much information is contributed by a token to a class of texts. The **mutual information** value will be equal to 0 if the token's distribution in all text classes is the same. The MI value approaches 1 as the token in any one class tends to occur more often in only that particular class of text. Exhibit 18 shows a simple depiction of some tokens with high MI scores for their corresponding text classes. Note how the tokens (or features) with the highest MI values narrowly relate to their corresponding text class name.

Exhibit 18 Tokens with Mutual Information (MI) Values for Two Given Text Classes

Text Classes: Sports or Politics			
Sports		Politics	
Token	MI Value	Token	MI Value
soccer	0.0781	election	0.0612
cup	0.0525	president	0.0511
match	0.0456	polls	0.0341
play	0.0387	vote	0.0288
game	0.0299	party	0.0202
team	0.0265	candidate	0.0201
win	0.0189	campaign	0.0201

Feature Engineering

As with structured data, feature engineering can greatly improve ML model training and remains a combination of art and science. The following are some techniques for feature engineering, which may overlap with text processing techniques.

1. *Numbers*: In text processing, numbers are converted into a token, such as "/number/." However, numbers can be of different lengths of digits representing different kinds of numbers, so it may be useful to convert different numbers into different tokens. For example, numbers with four digits may indicate years, and numbers with many digits could be an identification number. Four-digit numbers can be replaced with "/number4/," 10-digit numbers with "/number10/," and so forth.
2. *N-grams*: Multi-word patterns that are particularly discriminative can be identified and their connection kept intact. For example, "market" is a common word that can be indicative of many subjects or classes; the words "stock market" are used in a particular context and may be helpful to distinguish general texts from finance-related texts. Here, a bigram would be useful as it treats the two adjacent words as a single token (e.g., stock_market).
3. *Name entity recognition (NER)*: NER is an extensive procedure available as a library or package in many programming languages. The **name entity recognition** algorithm analyzes the individual tokens and their surrounding semantics while referring to its dictionary to tag an object class to the token. Exhibit 19 shows the NER tags of the text "*CFA Institute was formed in 1947 and is headquartered in Virginia.*" Additional object classes are, for example, MONEY, TIME, and PERCENT, which are not present in the example text. The NER tags, when applicable, can be used as features for ML model training for better model performance. NER tags can also help identify critical tokens on which such operations as lowercasing and stemming then can be avoided (e.g., Institute here refers to an organization rather than a verb). Such techniques make the features more discriminative.

Exhibit 19 Name Entity Recognition and Parts of Speech (POS) on Example Text

Token	NER Tag	POS Tag	POS Description
CFA	ORGANIZATION	NNP	Proper noun
Institute	ORGANIZATION	NNP	Proper noun
was		VBD	Verb, past tense
formed		VBN	Verb, past participle
in		IN	Preposition
1947	DATE	CD	Cardinal number
and		CC	Coordinating conjunction
is		VBZ	Verb, 3rd person singular present
headquartered		VBN	Verb, past participle

(continued)

Exhibit 19 (Continued)

Token	NER Tag	POS Tag	POS Description
in		IN	Preposition
Virginia	LOCATION	NNP	Proper noun

4 *Parts of speech (POS)*: Similar to NER, **parts of speech** uses language structure and dictionaries to tag every token in the text with a corresponding part of speech. Some common POS tags are noun, verb, adjective, and proper noun. Exhibit 19 shows the POS tags and descriptions of tags for the example text. POS tags can be used as features for ML model training and to identify the number of tokens that belong to each POS tag. If a given text contains many proper nouns, it means that it may be related to people and organizations and may be a business topic. POS tags can be useful for separating verbs and nouns for text analytics. For example, the word "market" can be a verb when used as "to market …" or noun when used as "in the market." Differentiating such tokens can help further clarify the meaning of the text. The use of "market" as a verb could indicate that the text relates to the topic of marketing and might discuss marketing a product or service. The use of "market" as a noun could suggest that the text relates to a physical or stock market and might discuss stock trading. Also for POS tagging, such compound nouns as "CFA Institute" can be treated as a single token. POS tagging can be performed using libraries or packages in programming languages.

In addition, many more creative techniques convey text information in a structured way to the ML training process. The goal of feature engineering is to maintain the semantic essence of the text while simplifying and converting it into structured data for ML.

EXAMPLE 4

Data Exploration

Paul Wang's analytics team at LendALot Corporation has completed its initial data preparation and wrangling related to their creditworthiness classification ML model building efforts. As a next step, Wang has asked one of the team members, Eric Kim, to examine the available structured data sources to see what types of exploratory data analysis might make sense. Kim has been tasked with reporting to the team on high-level patterns and trends in the data and which variables seem interesting. Greater situational awareness about the data can inform the team's decisions regarding model training and whether (and how) to incorporate textual big data in conjunction with the structured data inputs. Use the following sample of columns and rows Kim pulled for manual examination to answer the next questions.

1	ID	Loan Outcome	Income (USD)	Loan Amount (USD)	Credit Score	Loan Type	Free Responses to "Explain Credit Score" (excerpts from full text)
2	1	No Default	34,000	10,000	685	Mortgage	I am embarrassed that my score is below 700, but it was due to mitigating circumstances. I have developed a plan to improve my score.
3	2	No Default	63,050	49,000	770	Student Loan	I have a good credit score and am constantly looking to further improve it…
4	3	Defaulted	20,565	35,000	730	Student Loan	I think I have great credit. I don't think there are any issues. Having to provide a written response to these questions is kind of annoying…
5	4	No Default	50,021	10,000	664	Mortgage	I have a decent credit score. I regret not being as responsible in the past but feel I have worked hard to improve my score recently…
6	5	Defaulted	100,350	129,000	705	Car Loan	Honestly, my score probably would have been higher if I had worked harder. But it is probably good enough…
7	6	No Default	800,000	300,000	800	Boat Loan	I have worked hard to maintain a good credit rating. I am very responsible. I maintain a payment schedule and always stick to the payment plan…

1 Evaluate whether data visualization techniques, such as histograms, box plots, and scatterplots, could be relevant to exploratory data analysis.

2 State one visualization technique that could be used in relation to the free responses.

3 Describe how ranking methods can be used to select potentially interesting features to report back to the team.

4 State an example of a bigram from the free response texts that could be used to discriminate among loan outcomes.

Solution to 1:

The data provided include structured features (ID, Loan Outcome, Income, Loan Amount, Credit Score) and unstructured data. Histograms, box plots, and scatterplots are relevant visualization methods for structured data features. Histograms and box plots could be used by Kim to see how income, loan amount, and credit score are distributed. Moreover, these visualizations can be performed across all historical borrowing instances in the dataset as well as within the sets of defaulted loans versus non-defaulted loans. Scatterplots of income versus loan amount, income versus credit score, and loan amount versus credit score, both overall and within defaulted and non-defaulted datasets, can shed light on relationships between potentially important continuous variables.

Solution to 2:

For the text in the free response field, word clouds offer an appropriate starting point for exploratory analysis. A word cloud can enable a quick glimpse into the most frequently occurring words (i.e., term frequency). While some obvious words (e.g., "credit" and "score") may be valuable, other frequently occurring words (e.g., "worked," "hard," "probably," "embarrassed," "regret," "good," "decent," and "great") might have potential use for creditworthiness prediction.

Solution to 3:

Kim can use feature selection methods to rank all features. Since the target variable of interest (loan outcome) is discrete in this case, such techniques as chi-square and information gain would be well suited. These are univariate techniques that can score feature variables individually. In addition to the structured features, these univariate ranking methods can also be applied to word count-related features, such as term frequency and document frequency, that are derived from the text using frequently occurring words. Such frequently occurring words (e.g., “worked” and “hard”) can be identified from the word cloud.

Solution to 4:

The bigrams “credit_score” and “worked_hard” from the text in the free response section may have potential to discriminate among loan outcomes.

EXAMPLE 5

Textual Feature Representations for ML Model Building

Having completed their exploration of the data, Paul Wang’s analytics team at LendALot Corporation recognizes the importance of incorporating features derived from text data in their ML models for classifying creditworthiness. Wang has asked his colleagues, Lynn Lee and Eric Kim, to propose textual feature representations that might be well suited to constructing features for their task. As a starting point, Lee and Kim review the following sample of data:

1	ID	Loan Outcome	Income (USD)	Loan Amount (USD)	Credit Score	Loan Type	Free Responses to “Explain Credit Score” (excerpts from full text)
2	1	No Default	34,000	10,000	685	Mortgage	I am embarrassed that my score is below 700, but it was due to mitigating circumstances. I have developed a plan to improve my score.
3	2	No Default	63,050	49,000	770	Student Loan	I have a good credit score and am constantly looking to further improve it…
4	3	Defaulted	20,565	35,000	730	Student Loan	I think I have great credit. I don’t think there are any issues. Having to provide a written response to these questions is kind of annoying…
5	4	No Default	50,021	10,000	664	Mortgage	I have a decent credit score. I regret not being as responsible in the past but feel I have worked hard to improve my score recently…
6	5	Defaulted	100,350	129,000	705	Car Loan	Honestly, my score probably would have been higher if I had worked harder. But it is probably good enough…
7	6	No Default	800,000	300,000	800	Boat Loan	I have worked hard to maintain a good credit rating. I am very responsible. I maintain a payment schedule and always stick to the payment plan…

Based on the information given, address the following questions.

1 Describe three textual feature representations that Lee and Kim should consider for their text data.

2 Describe a rationale for adopting each of the three textual feature representations identified in Question 1.

Solution 1:

Lee and Kim should consider bag-of-words (BOW), n-grams, and parts-of-speech (POS) as key textual feature representations for their text data. Conversely, name entity recognition (NER) might not be as applicable in this context because the data on prospective borrowers does not include any explicit references to people, locations, dates, or organizations.

Solution 2:

All three textual feature representations have the potential to add value.

Bag-of-words (BOW) is typically applicable in most contexts involving text features derived from languages where token boundaries are explicitly present (e.g., English) or can be inferred through processing (e.g., a different language, such as Spanish). BOW is generally the best starting point for most projects exploring text feature representations.

N-grams, representations of word or token sequences, are also applicable. N-grams can offer invaluable contextual information that can complement and enrich a BOW. In this specific credit-worthiness context, we examine the BOW token "worked." It appears three times (rows 5–7), twice in no-default loan texts and once in a defaulted loan text. This finding suggests that "worked" is being used to refer to the borrower's work ethic and may be a good predictor of credit worthiness. Digging deeper and looking at several trigrams (i.e., three-token sequences) involving "worked," we see that "have_worked_hard" appears in the two no-default loan related texts (referring to borrower accomplishments and plans) and "had_worked_harder" appears in the defaulted loan text (referring to what could have been done). This example illustrates how n-grams can provide richer contextualization capabilities for the creditworthiness prediction ML models.

Parts-of-speech tags can add value because they identify the composition of the texts. For example, POS provides information on whether the prospective borrowers are including many action words (verbs) or descriptors (adjectives) and whether this is being done differently in instances of no-default versus instances of defaulted loans.

MODEL TRAINING 6

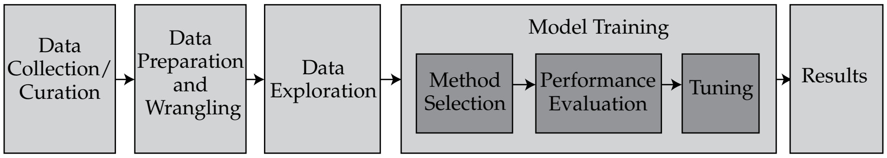
Machine learning model training is a systematic, iterative, and recursive process. The number of iterations required to reach optimum results depends on:

- the nature of the problem and input data and
- the level of model performance needed for practical application.

Machine learning models combine multiple principles and operations to provide predictions. As seen in the last two sections, typical ML model building requires data preparation and wrangling (cleansing and preprocessing) and data exploration (exploratory data analysis as well as feature selection and engineering). In addition, domain knowledge related to the nature of the data is required for good model building and training. For instance, knowledge of investment management and securities trading is important when using financial data to train a model for predicting costs of trading stocks. It is crucial for ML engineers and domain experts to work together in building and training robust ML models.

The three tasks of ML model training are method selection, performance evaluation, and tuning. Exhibit 20 outlines model training and its three component tasks. Method selection is the art and science of deciding which ML method(s) to incorporate and is guided by such considerations as the classification task, type of data, and size of data. Performance evaluation entails using an array of complementary techniques and measures to quantify and understand a model's performance. Tuning is the process of undertaking decisions and actions to improve model performance. These steps may be repeated multiple times until the desired level of ML model performance is attained. Although no standard rulebook for training an ML model exists, having a fundamental understanding of domain-specific training data and ML algorithm principles plays a vital role in good model training.

Exhibit 20 Model Training Stage



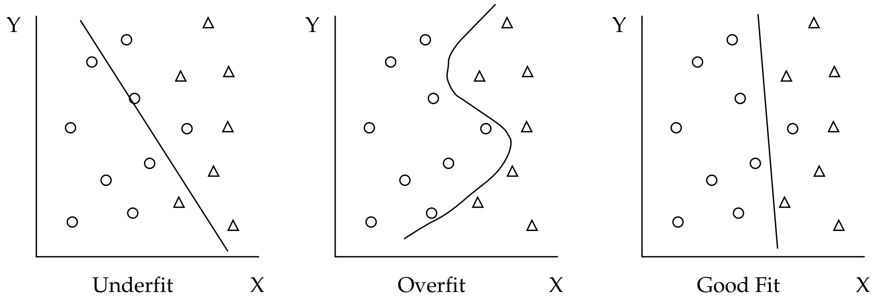
Before training a model, it is important to state the problem, define objectives, identify useful data points, and conceptualize the model. Conceptualization is like a blueprint on a drawing board, a modifiable plan that is necessary to initiate the model training process. Because modeling is an iterative process, many changes and refinements will be made to the model plan as the process evolves.

6.1 Structured and Unstructured Data

The ML model training process for structured and unstructured data is typically the same. Most ML models are intended to train on structured data, so unstructured data in the data preparation stage are processed and organized into a structured format. The systematic processing of unstructured text data so that they can be structured in the form of a data matrix has been previously covered. Similarly, other forms of unstructured data can also be prepared and formed into data matrixes or tables for ML training.

The fundamental idea of ML model training is fitting a system of rules on a training dataset to reveal a pattern in the data. In other words, fitting describes the degree to which (or how well) an ML model can be generalized to new data. A good model fit results in good model performance and can be validated using new data outside of the training dataset (i.e., out-of-sample). Exhibit 21 shows model decision boundaries in three possible model fitting scenarios for a classification task comprising two different classes of data (i.e., circles and triangles). The model on the left is underfit; it does not fit the training data well enough since it results in four misclassification errors (three circles and one triangle). Although the center model that generates the "S"-shaped line has the best accuracy (no errors) on the training data, it is overfit (i.e., fits the training data too well) and thus unlikely to perform well on future test cases. The model on the right (with one classification error, a circle) is a model with good fit (i.e., it fits the training data well but not so well that it cannot be generalized to out-of-sample data).

Exhibit 21 Model Fitting Scenarios: Underfit, Overfit, and Good Fit



Model fitting errors are caused by several factors—the main ones being dataset size and number of features in the dataset.

- *Dataset Size*: Small datasets can lead to underfitting of the model since small datasets often are not sufficient to expose patterns in the data. Restricted by a small dataset, an ML model may not recognize important patterns.
- *Number of Features*: A dataset with a small number of features can lead to underfitting, and a dataset with a large number of features can lead to overfitting. As with small dataset size, a small number of features may not carry all the characteristics that explain relationships between the target variable and the features. Conversely, a large number of features can complicate the model and potentially distort patterns in the data due to low degrees of freedom, causing overfitting. Therefore, appropriate feature selection using the types of techniques described earlier (e.g., chi-square, mutual information) is a key factor in minimizing such model overfitting.

Feature engineering tends to prevent underfitting in the training of the model. New features, when engineered properly, can elevate the underlying data points that better explain the interactions of features. Thus, feature engineering can be critical to overcome underfitting. Method-related factors that affect model fitting are explained shortly under tuning.

Method Selection

ML model training is a craft (part art and part science); it has no strict guidelines. Selecting and applying a method or an algorithm is the first step of the training process. Method selection is governed by the following factors:

1 *Supervised or unsupervised learning*. The data for training and testing supervised ML models contain **ground truth**, the known outcome (i.e., target variable) of each observation in these datasets. Unsupervised ML modeling is relatively challenging because of the absence of ground truth (i.e., no target variable). Supervised models bring a structure that may or may not be supported by the data. Unsupervised models bring no structure beyond that which arises from the given data. For supervised learning (with labeled training data), typical methods of choice are regression, ensemble trees, support vector machines (SVMs), and neural networks (NNs). Supervised learning would be used, for example, for default prediction based on high-yield corporate bond issuer data. For unsupervised learning, common methods are dimensionality

reduction, clustering, and anomaly detection. Unsupervised learning, for example, would be used for clustering financial institutions into different groups based on their financial attributes.

2 *Type of data.* For numerical data (e.g., predicting stock prices using historical stock market values), classification and regression tree (CART) methods may be suitable. For text data (for example, predicting the topic of a financial news article by reading the headline of the article), such methods as generalized linear models (GLMs) and SVMs are commonly used. For image data (e.g., identifying objects in a satellite image, such as tanker ships moving in and out of port), NNs and deep learning methods tend to perform better than others. For speech data (e.g., predicting financial sentiment from quarterly earnings' conference call recordings), deep learning methods can offer promising results.

3 *Size of data.* A typical dataset has two basic characteristics: number of instances (i.e., observations) and number of features. The combination of these two characteristics can govern which method is most suitable for model training. For instance, SVMs have been found to work well on "wider" datasets with 10,000 to 100,000 features and with fewer instances. Conversely, NNs often work better on "longer" datasets, where the number of instances is much larger than the number of features.

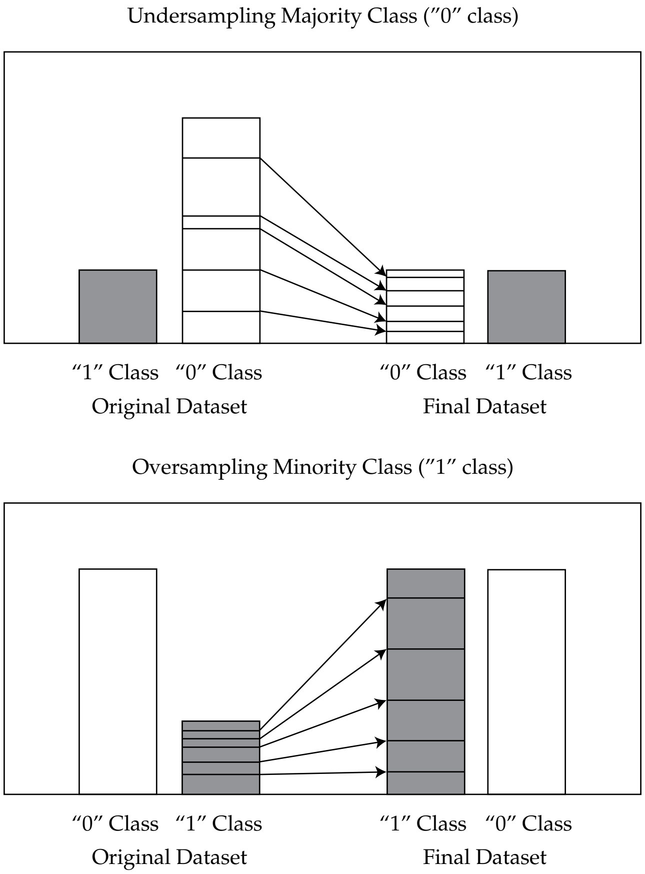
Once a method is selected, certain method-related decisions (e.g., on hyperparameters) need to be made. These decisions include the number of hidden layers in a neural network and the number of trees in ensemble methods (discussed later in the sub-section on tuning). In practice, datasets can be a combination of numerical and text data. To deal with mixed data, the results from more than one method can be combined. Sometimes, the predictions from one method can be used as predictors (features) by another. For example, unstructured financial text data can be used with logistic regression to classify stock sentiment as either positive or negative. Then, this sentiment classification cam be used as a predictor in a larger model, say CART, that also uses structured financial data as predictors for the purpose of stock selection. Finally, more than one method can be used and the results combined with quantitative or subjective weighing to exploit the advantages of each method.

Before model training begins, in the case of supervised learning the master dataset is split into three subsets used for model training and testing purposes. The first subset, a training set used to train the model, should constitute approximately 60% of the master dataset. The second subset, a cross-validation set (or validation set) used to tune and validate the model, should constitute approximately 20% of the master dataset. The third subset is a test set for testing the model and uses the remaining data. The data are split using a random sampling technique, such as the k-fold method. A commonly recommended split ratio is 60:20:20, as detailed above; however, the split percentages can vary. For unsupervised learning, no splitting is needed due to the absence of labeled training data.

Class imbalance, where the number of instances for a particular class is significantly larger than for other classes, may be a problem for data used in supervised learning because the ML classification method's objective is to train a high-accuracy model. In a high-yield bond default prediction example, say for corporate issuers in the BB+/Ba1 to B+/B1 credit quality range, issuers who defaulted (positive or "1" class) would be very few compared to issuers who did not default (negative or "0" class). Hence, on such training data, a naive model that simply assumes no corporate issuer will default may achieve good accuracy—albeit with all default cases misclassified. Balancing the training data can help alleviate such problems. In cases of unbalanced data, the "0" class (majority class) can be randomly undersampled or the "1" class (minority class) randomly oversampled. The random sampling can be done with or without replacement because they both work the same in general probability theory.

Exhibit 22 depicts the idea of undersampling of the majority class and oversampling of the minority class. In practice, the choice of whether to undersample or oversample depends on the specific problem context. Advanced techniques can also reproduce synthetic observations from the existing data, and the new observations can be added to the dataset to balance the minority class.

Exhibit 22 Undersampling and Oversampling



Performance Evaluation

It is important to measure the model training performance or goodness of fit for validation of the model. We shall cover several techniques to measure model performance that are well suited specifically for binary classification models.

1 *Error analysis.* For classification problems, error analysis involves computing four basic evaluation metrics: true positive (TP), false positive (FP), true negative (TN), and false negative (FN) metrics. FP is also called a Type I error, and FN is also called a Type II error. Exhibit 23 shows a **confusion matrix**, a grid that is used to summarize values of these four metrics.

Exhibit 23 Confusion Matrix for Error Analysis

		Actual Training Labels	
		Class “1”	Class “0”
Predicted Results	Class “1”	True Positives (TP)	False Positives (FP) Type I Error
	Class “0”	False Negatives (FN) Type II Error	True Negatives (TN)

Additional metrics, such as precision and recall, can be computed. Assume in the following explanation that Class “0” is “not defective” and Class “1” is “defective.” **Precision** is the ratio of correctly predicted positive classes to all predicted positive classes. Precision is useful in situations where the cost of FP, or Type I error, is high—or example, when an expensive product fails quality inspection (predicted Class “1”) and is scrapped, but it is actually perfectly good (actual Class “0”). **Recall** (also known as *sensitivity*) is the ratio of correctly predicted positive classes to all actual positive classes. Recall is useful in situations where the cost of FN or Type II error is high—for example, when an expensive product passes quality inspection (predicted Class “0”) and is sent to the valued customer, but it is actually quite defective (actual Class “1”). The formulas for precision and recall are:

$$\text{Precision (P)} = \text{TP}/(\text{TP} + \text{FP}). \quad (3)$$

$$\text{Recall (R)} = \text{TP}/(\text{TP} + \text{FN}). \quad (4)$$

Trading off precision and recall is subject to business decisions and model application. Therefore, additional evaluation metrics that provide the overall performance of the model are generally used. The two overall performance metrics are accuracy and F1 score. **Accuracy** is the percentage of correctly predicted classes out of total predictions. **F1 score** is the harmonic mean of precision and recall. F1 score is more appropriate (than accuracy) when unequal class distribution is in the dataset and it is necessary to measure the equilibrium of precision and recall. High scores on both of these metrics suggest good model performance. The formulas for accuracy and F1 score are as follows:

$$\text{Accuracy} = (\text{TP} + \text{TN})/(\text{TP} + \text{FP} + \text{TN} + \text{FN}). \quad (5)$$

$$\text{F1 score} = (2 * \text{P} * \text{R})/(\text{P} + \text{R}). \quad (6)$$

Exhibit 24 illustrates computations of model evaluation metrics and performance scores on a sample dataset.

Exhibit 24 Performance Metrics and Scores Computation

Sample Dataset with Classification Results

Observation	Actual Training Labels	Predicted Results	Classification
1	1	1	TP
2	0	0	TN
3	1	1	TP
4	1	0	FN
5	1	1	TP
6	1	0	FN
7	0	0	TN
8	0	0	TN
9	0	0	TN
10	0	1	FP

Confusion Matrix

		Actual Training Labels	
		Class "1"	Class "0"
Predicted Results	Class "1"	3 (TP)	1 (FP)
	Class "0"	2 (FN)	4 (TN)

Performance Metrics

$$TP = 3, FP = 1, FN = 2, TN = 4$$

$$P = 3 / (3+1) = 0.75$$

$$R = 3 / (3+2) = 0.60$$

$$\text{F1 Score} = (2 \times 0.75 \times 0.60) / (0.75 + 0.60) = 0.67$$

$$\text{Accuracy} = (3 + 4) / (3 + 1 + 4 + 2) = 0.70$$

In Exhibit 24, if all "1" classes were predicted correctly (no FPs), the precision would have been equal to 1. If all "0" classes were predicted correctly (no FNs), the recall would have been equal to 1. Thus, the resulting F1 score would have been equal to 1. The precision of 0.75 and recall of 0.60 indicate that the model is better at minimizing FPs than FNs. To find the equilibrium between precision and recall, F1 score is calculated, which is equal to 0.67. The F1 score is closer to the smaller value among both precision and recall, giving the model a more appropriate score rather than just an arithmetic mean. Accuracy, the percentage of correct predictions (for both classes) made by the model, is equal to 0.70. Accuracy would be equal to 1 if all predictions were correct. As the number of "1" and "0" classes is equal in the dataset (i.e., a balanced dataset), accuracy can be considered an appropriate performance measure in this case. If the number of classes in a dataset is unequal; however, then F1 score should be used as the overall performance measure for the model.

2 *Receiver Operating Characteristic (ROC).* This technique for assessing model performance involves the plot of a curve showing the trade-off between the false positive rate (x-axis) and true positive rate (y-axis) for various cutoff points—for example, for the predicted probability (p) in a logistic regression. The formulas for false positive rate and true positive rate (note that true positive rate is the same as recall) are:

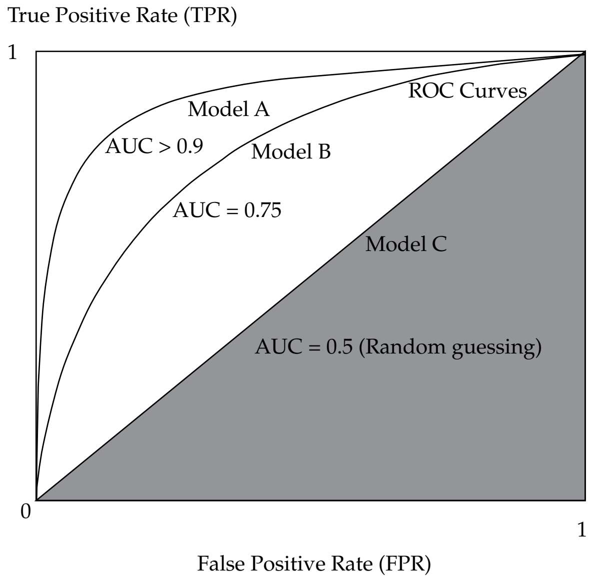
$$\text{False positive rate (FPR)} = FP/(TN + FP) \text{ and} \tag{7}$$

$$\text{True positive rate (TPR)} = TP/(TP + FN). \tag{8}$$

If p from a logistic regression model for a given observation is greater than the cutoff point (or threshold), then the observation is classified as class = 1. Otherwise, the observation will be classified as class = 0.

The shape of the ROC curve provides insight into the model's performance. A more convex curve indicates better model performance. Area under the curve (AUC) is the metric that measures the area under the ROC curve. An AUC close to 1.0 indicates near perfect prediction, while an AUC of 0.5 signifies random guessing. Exhibit 25 displays three ROC curves and indicates their respective AUC values. It is clear from observing the shapes of the ROC curves and their AUCs that Model A—with the most convex ROC curve with AUC of more than 0.9 (or 90%)—is the best performing among the three models.

Exhibit 25 ROC Curves and AUCs



3 *Root Mean Squared Error (RMSE).* This measure is appropriate for continuous data prediction and is mostly used for regression methods. It is a single metric that captures all the prediction errors in the data (*n*). The root mean squared error is computed by finding the square root of the mean of the squared differences between the actual values and the model's predicted values (error). A small RMSE indicates potentially better model performance. The formula for RMSE is:

$$\text{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(\text{Predicted}_i - \text{Actual}_i)^2}{n}} \quad (9)$$

Tuning

Once the model is evaluated, certain decisions and actions must be taken based on the findings to improve the performance of the model. If the prediction error on the training set is high, the model is underfitting. If the prediction error on the cross-validation (CV) set is significantly higher than on the training set, the model is overfitting. Model fitting has two types of error: bias and variance. Bias error is associated with underfitting, and variance error is associated with overfitting. Bias error is high when a model is overly simplified and does not sufficiently learn from the patterns in the training data. Variance error is high when the model is overly complicated and memorizes the training data so much that it will likely perform poorly on new data. It is not possible to completely eliminate both types of errors. However, both errors can be minimized so the total aggregate error (bias error + variance error) is at a minimum. The bias–variance trade-off is critical to finding an optimum balance where a model neither underfits nor overfits.

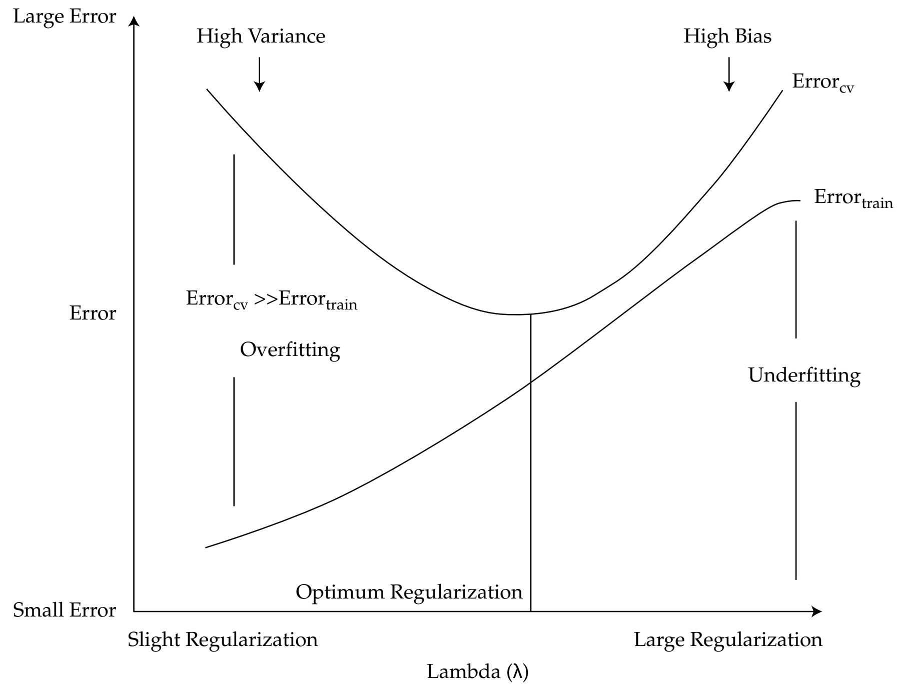
1 *Parameters* are critical for a model and are dependent on the training data. Parameters are learned from the training data as part of the training process by an optimization technique. Examples of parameters include coefficients in regression, weights in NN, and support vectors in SVM.

2 *Hyperparameters* are used for estimating model parameters and are not dependent on the training data. Examples of hyperparameters include the regularization term (λ) in supervised models, activation function and number of hidden layers in NN, number of trees and tree depth in ensemble methods, *k* in k-nearest neighbor classification and k-means clustering, and p-threshold in logistic regression. Hyperparameters are manually set and tuned.

For example, if a researcher is using a logistic regression model to classify sentences from financial statements into positive or negative stock sentiment, the initial cutoff point for the trained model might be a p-threshold of 0.50 (50%). Therefore, any sentence for which the model produces a probability >50% is classified as having positive sentiment. The researcher can create a confusion matrix from the classification results (of running the CV dataset) to determine such model performance metrics as accuracy and F1 score. Next, the researcher can vary the logistic regression's p-threshold—say to 0.55 (55%), 0.60 (60%), or even 0.65 (65%)—and then re-run the CV set, create new confusion matrixes from the new classification results, and compare accuracy and F1 scores. Ultimately, the researcher would select the logistic regression model with a p-threshold value that produces classification results generating the highest accuracy and F1 scores. Note that the process just outlined will be demonstrated in Section 7.

There is no general formula to estimate hyperparameters. Thus, tuning heuristics and such techniques as grid search are used to obtain the optimum values of hyperparameters. **Grid search** is a method of systematically training an ML model by using various combinations of hyperparameter values, cross validating each model, and determining which combination of hyperparameter values ensures the best model performance. The model is trained using different combinations of hyperparameter values until the optimum set of values are found. Optimum values must result in similar performance of the model on training and CV datasets, meaning that the training error and CV error are close. This ensures that the model can be generalized to test data or to new data and thus is less likely to overfit. The plot of training errors for each value of a hyperparameter (i.e., changing model complexity) is called a fitting curve. Fitting curves provide visual insight on the model's performance (for the given hyperparameter and level of model complexity) on the training and CV datasets and are visually helpful to tune hyperparameters. Exhibit 26 shows the bias–variance error trade-off by plotting a generic fitting curve for a regularization hyperparameter (λ).

Exhibit 26 Fitting Curve for Regularization Hyperparameter (λ)



Slight regularization lightly penalizes model complexity, thereby allowing most or all of the features to be included in the model and thus potentially enabling the model to "memorize" the data. Typically with no or slight regularization, the prediction error on the training dataset is small while the prediction error on the CV dataset is significantly larger. This difference in error is variance. High variance error, which typically results from too many features and model complexity, results in model overfitting. When high variance error and low bias error exist, the model performs well on the training dataset but generates many FP and FN errors on the CV dataset; in other words, the model is overfitted and does not generalize to new data well.

Large regularization excessively penalizes model complexity, thereby allowing too few of the features to be included in the model and causing the model to learn less from the data. The model may lack the necessary predictor variables and complexity needed to discern underlying patterns in the data. Typically with large regularization, the prediction errors on the training and CV datasets are both large. Large prediction errors on the training dataset indicate high bias, and high bias error results from model underfitting. When high bias error exists, the model does not perform well on either training or CV datasets because it is typically lacking important predictor variables.

Optimum regularization minimizes both variance and bias errors in a balanced fashion. It penalizes model complexity just enough so that only the most important features are included in the model. This process prevents the model from memorizing the data while enabling the model to learn enough from the data to distinguish important patterns. This results in prediction errors in both training and CV datasets that are similar and also minimal. The range of optimum regularization values can be found heuristically using such techniques as grid search.

If high bias or variance exists after the tuning of hyperparameters, either a larger number of training examples (instances) may be needed or the number of features included in the model may need to be decreased (in the case of high variance) or increased (in the case of high bias). The model then needs to be re-trained and re-tuned using the new training dataset. In the case of a complex model, where a large model is comprised of sub-model(s), ceiling analysis can be performed. **Ceiling analysis** is a systematic process of evaluating different components in the pipeline of model building. It helps to understand what part of the pipeline can potentially improve in performance by further tuning. For example, a stock market prediction model needs historical data from the stock market and perhaps news articles related to the stocks. The sub-model will extract relevant information from the news articles or classify the sentiment of the news articles. The results of the sub-model will feed into the larger model as features. Thus, the performance of the larger model depends on performance of the sub-model(s). Ceiling analysis can help determine which sub-model needs to be tuned to improve the overall accuracy of the larger model.

FINANCIAL FORECASTING PROJECT: CLASSIFYING AND PREDICTING SENTIMENT FOR STOCKS

7

Robo-readers are automated programs used to analyze large quantities of text, including news articles and social media. In the financial services space, robo-readers are being used by investors to examine how views expressed in text relate to future company performance. One important dimension that robo-readers look to analyze is sentiment polarity—which means how positive, negative, or neutral a particular phrase or statement is regarding a "target." For example, in the statement "XYZ Corporation is doing terrific things with its new product innovation," positive sentiment (i.e., the polarity) is being expressed regarding XYZ Corporation (i.e., the target of the sentiment). Such sentiment can provide invaluable predictive power, both alone and when coupled with structured financial data, for predicting stock price movements for individual firms and for portfolios of companies.

To provide a practical application, we use a financial forecasting project to examine how effectively sentiment—expressed in English news articles on LexisNexis (a searchable database of news articles) related to all companies listed on the NASDAQ OMX Helsinki (Finland)—can be classified. To accomplish this task, we followed the text ML model building steps presented in Sections 3 to 6 of this reading.

7.1 Text Curation, Preparation, and Wrangling

Text Curation

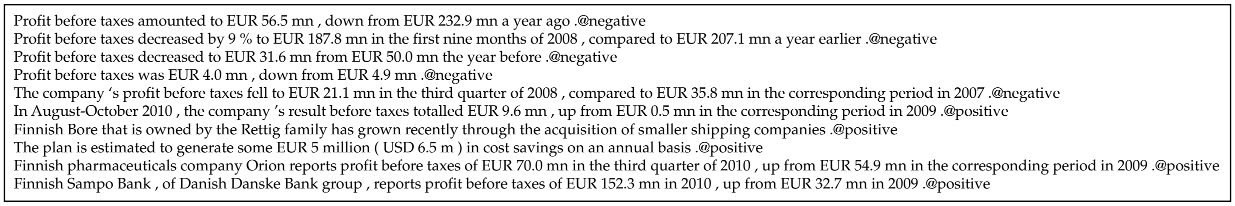
The text data used in this financial forecasting project are a collection of English language sentences from financial and economic news sources. The text data are acquired from the Financial Phrase Bank located on the website Researchgate.net.[2] The compressed folder contains six text files. The first two files are license and readme files. The other four files contain the text data. The data are presented in a text document format (.txt), which can be opened and viewed using any text editor. Note that this is cross-sectional data (not time series data).

A total of 14,780 sentences are in the four files. The sentiment of each sentence has already been labeled with one of three sentiment classes: positive, neutral, or negative. The sentiment classes are provided from an investor's perspective and may be useful for predicting whether a sentence may have a corresponding positive, neutral, or negative influence on the respective company's stock price.

[2] https://www.researchgate.net/publication/251231364_FinancialPhraseBank-v10.

This project uses sentences from two of the text files (Sentences_AllAgree and Sentences_75Agree), labeled as either in the positive or negative sentiment class, for a total of 2,180 sentences. There are 1,457 positive sentiment class sentences and 723 negative sentiment class sentences. A supervised ML model is trained, validated, and tested using these data. The final ML model can be used to predict the sentiment classes of sentences present in similar financial news statements. Exhibit 27 shows a sample of 10 rows of raw text from the Sentences_AllAgree text file. Note the sentiment annotations at the end of each sentence with prefix character "@."

Exhibit 27 Ten Sample Sentences and Sentiment from Raw Text File (Sentences_AllAgree.txt)



```
Profit before taxes amounted to EUR 56.5 mn , down from EUR 232.9 mn a year ago .@negative
Profit before taxes decreased by 9 % to EUR 187.8 mn in the first nine months of 2008 , compared to EUR 207.1 mn a year earlier .@negative
Profit before taxes decreased to EUR 31.6 mn from EUR 50.0 mn the year before .@negative
Profit before taxes was EUR 4.0 mn , down from EUR 4.9 mn .@negative
The company 's profit before taxes fell to EUR 21.1 mn in the third quarter of 2008 , compared to EUR 35.8 mn in the corresponding period in 2007 .@negative
In August-October 2010 , the company 's result before taxes totalled EUR 9.6 mn , up from EUR 0.5 mn in the corresponding period in 2009 .@positive
Finnish Bore that is owned by the Rettig family has grown recently through the acquisition of smaller shipping companies .@positive
The plan is estimated to generate some EUR 5 million ( USD 6.5 m ) in cost savings on an annual basis .@positive
Finnish pharmaceuticals company Orion reports profit before taxes of EUR 70.0 mn in the third quarter of 2010 , up from EUR 54.9 mn in the corresponding period in 2009 .@positive
Finnish Sampo Bank , of Danish Danske Bank group , reports profit before taxes of EUR 152.3 mn in 2010 , up from EUR 32.7 mn in 2009 .@positive
```

Text Preparation (Cleansing)

The raw text data (i.e., sentences) are initially organized into a data table. The data table contains two columns: The first column (sentence) is for the text, and the second column (sentiment) is for the corresponding sentiment class. The separator character, which is "@" in this case, is used to split the data into text and sentiment class columns. A collection of text data in any form, including list, matrix, or data table forms, is called a **corpus**. Exhibit 28 shows a sample of 10 sentences from the data table corpus.

Exhibit 28 Ten Sample Rows of the Data Table (Corpus)

Sentence	Sentiment
Profit before taxes amounted to EUR 56.5 mn , down from EUR 232.9 mn a year ago .	negative
Profit before taxes decreased by 9 % to EUR 187.8 mn in the first nine months of 2008 , compared to EUR 207.1 mn a year earlier .	negative
Profit before taxes decreased to EUR 31.6 mn from EUR 50.0 mn the year before .	negative
Profit before taxes was EUR 4.0 mn , down from EUR 4.9 mn .	negative
The company 's profit before taxes fell to EUR 21.1 mn in the third quarter of 2008 , compared to EUR 35.8 mn in the corresponding period in 2007 .	negative
In August-October 2010 , the company 's result before taxes totalled EUR 9.6 mn , up from EUR 0.5 mn in the corresponding period in 2009 .	positive
Finnish Bore that is owned by the Rettig family has grown recently through the acquisition of smaller shipping companies .	positive
The plan is estimated to generate some EUR 5 million (USD 6.5 m) in cost savings on an annual basis .	positive
Finnish pharmaceuticals company Orion reports profit before taxes of EUR 70.0 mn in the third quarter of 2010 , up from EUR 54.9 mn in the corresponding period in 2009 .	positive
Finnish Sampo Bank , of Danish Danske Bank group , reports profit before taxes of EUR 152.3 mn in 2010 , up from EUR 32.7 mn in 2009 .	positive

The raw text contains punctuations, numbers, and white spaces that may not be necessary for model training. Text cleansing involves removing, or incorporating appropriate substitutions for, potentially extraneous information present in the text. Operations to remove html tags are unnecessary because none are present in the text

Punctuations: Before stripping out punctuations, percentage and dollar symbols are substituted with word annotations to retain their essence in the financial texts. Such word annotation substitutions convey that percentage and currency-related tokens were involved in the text. As the sentences have already been identified within and extracted from the source text, punctuation helpful for identifying discrete sentences—such as periods, semi-colons, and commas—are removed. Some special characters, such as "+" and "©," are also removed. It is a good practice to implement word annotation substitutions before removing the rest of the punctuations.

Numbers: Numerical values of numbers in the text have no significant utility for sentiment prediction in this project because sentiment primarily depends on the words in a sentence. Here is an example sentence: "*Ragutis, which is based in Lithuania's second-largest city, Kaunas, boosted its sales last year 22.3 percent to 36.4 million litas*." The word "boosted" implies that there was growth in sales, so analysis of this sentiment does not need to rely on interpretation of numerical text data. Sentiment analysis typically does not involve extracting, interpreting, and calculating relevant numbers but instead seeks to understand the context in which the numbers are used. Other commonly occurring numbers are dates and years, which are also not required to predict sentence sentiment. Thus, all numbers present in the text are removed for this financial sentiment project. However, prior to removing numbers, abbreviations representing orders of magnitude, such as million (commonly represented by "m," "mln," or "mn"), billion, or trillion, are replaced with the complete word. Retaining these orders of magnitude-identifying words in the text preserves the original text meaning and can be useful in predicting sentence sentiment.

Whitespaces: White spaces are present in the raw text. Additional white spaces occur after performing the above operations to remove extraneous characters. The white spaces must be removed to keep the text intact. Exhibit 29 shows the sample text after cleansing. The cleansed text is free of punctuations and numbers, with useful substitutions.

Exhibit 29 Ten Sample Rows After Cleansing Process

Sentence	Sentiment
Profit before taxes amounted to EUR million down from EUR million a year ago	negative
Profit before taxes decreased by percentSign to EUR million in the first nine months of compared to EUR million a year earlier	negative
Profit before taxes decreased to EUR million from EUR million the year before	negative
Profit before taxes was EUR million down from EUR million	negative
The companys profit before taxes fell to EUR million in the third quarter of compared to EUR million in the corresponding period in	negative
In August October the companys result before taxes totalled EUR million up from EUR million in the corresponding period in	positive
Finnish Bore that is owned by the Rettig family has grown recently through the acquisition of smaller shipping companies	positive
The plan is estimated to generate some EUR million USD million in cost savings on an annual basis	positive

(continued)

Exhibit 29 (Continued)

Sentence	Sentiment
Finnish pharmaceuticals company Orion reports profit before taxes of EUR million in the third quarter of up from EUR million in the corresponding period in	positive
Finnish Sampo Bank of Danish Danske Bank group reports profit before taxes of EUR million in up from EUR million in	positive

Text Wrangling (Preprocessing)

The cleansed text needs to be normalized using the following normalization procedures:

1 *Lowercasing* of all text to consolidate duplicate words (example, "THE," "The," and "the").

2 *Stop words* are not removed because some stop words (e.g., not, more, very, and few) carry significant meaning in the financial texts that is useful for sentiment prediction. Some stop words, such as articles (a, an, the), may be removed. Nevertheless, to avoid confusion no words are removed at this point. This issue will be revisited during the data exploration stage, which will carefully examine the text using frequency analysis and find custom stop words (common words) for these particular text data.

3 *Stemming*, the converting of inflected forms of a word into its base word (stem), is performed on the text as it is simple to perform and is appropriate for training an ML model for sentiment prediction.

White spaces are stripped after performing these operations. As part of text normalization, different currency abbreviations, such as EUR and USD, can be converted into a single token, such as "currencysign." As we are dealing with financial domain text, the earlier substitution of dollarsign can be replaced with currencysign as well. This step will remove tokens that are different but redundant in nature while maintaining their meaning. Through careful examination of the text and use of domain knowledge, similar substitutions of redundant tokens can be performed. Exhibit 30 shows how the sample text appears after normalization.

Exhibit 30 Ten Sample Rows After Normalization Process

Sentence	Sentiment
profit befor tax amount to currencysign million down from currencysign million a year ago	negative
profit befor tax decreas by percentsign to currencysign million in the first nine month of compar to currencysign million a year earlier	negative
profit before tax decreas to currencysign million from currencysign million the year befor	negative
profit befor tax was currencysign million down from currencysign million	negative
the compani profit befor tax fell to currencysign million in the third quarter of compar to currencysign million in the correspond period in	negative
in august octob the compani result befor tax total currencysign million up from currencysign million in the correspond period in	positive
finnish bore that is own by the rettig famili has grown recent through the acquisit of smaller shipping company	positive
the plan is estim to generat some currencysign million currencysign million in cost save on an annual basi	positive

Exhibit 30 (Continued)

Sentence	Sentiment
finnish pharmaceut compani orion report profit befor tax of currencysign million in the third quarter of up from currencysign million in the correspond period in	positive
finnish sampo bank of danish danske bank group report profit befor tax of currencysign million in up from currencysign million in	positive

The normalized text is tokenized, resulting in 2,673 unique tokens. Altogether, these unique tokens comprise the bag-of-words (BOW) of the text corpus. Exhibit 31 shows a sample of 100 tokens from the BOW. This preliminary unigram BOW can be used to construct a document term matrix (DTM) for ML training.

Exhibit 31 One Hundred Sample Tokens from Preliminary Unigram BOW

“for”
“currencysign”
“move”
“percentsign”
“report”
“prior”
“led”
“handset”
“board”
“user”
“the”
“million”
“zero”
“and”
“cloth”
“foundri”
“nokia”
“nokian”
“director”
“as”
“last”
“from”
“pre”
“oper”
“retail”
“divis”
“after”
“tyre”
“will”
“decemb”
“quarter”
“same”
“tax”
“rose”
“chain”
“machin”
“announc”
“fourth”
“propos”
“market”
“of”
“period”
“profit”
“correspond”
“sepp”
“shop”
“plan”
“earn”
“dividned”
“number”
“componenta”
“a”
“loss”
“repres”
“is”
“helsinki”
“team”
“beat”
“per”
“custom”
“net”
“year”
“in”
“total”
“consolid”
“afx”
“with”
“analyst”
“paid”
“late”
“sale”
“earlier”
“third”
“up”
“reach”
“share”
“sanyo”
“expect”
“megafon”
“accord”
“doubl”
“while”
“increas”
“finnish”
“amount”
“close”
“manufactur”
“dealer”
“subscrib”
“teliasonera”
“to”
“it”
“by”
“talentum”
“compar”
“higher”
“g”
“said”
“base”
“estim”

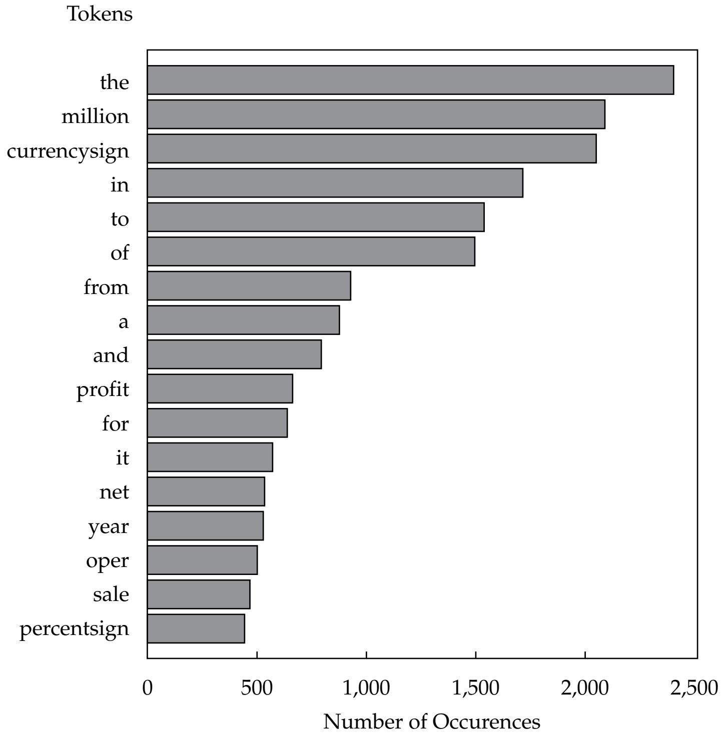
The final DTM for ML model training will be prepared after the data exploration stage. Data exploration may reveal unnecessary tokens or anomalies in the data. Any unnecessary tokens that are not informative must be removed, which will also impact the creation of n-grams. Thus, the final DTM must be made after further analyses and operations, such as exploratory data analysis and feature selection.

7.2 Data Exploration

Exploratory Data Analysis

Exploratory data analysis (EDA) performed on text data provides insights on word distribution in the text. Word counts from all the sentences are computed. These word counts can be used to examine outlier tokens—words that are most commonly and least commonly present in the texts. The most frequent word occurrences in all sentences from the dataset are shown in Exhibit 32. These common words will be removed during the feature selection step. Notably, the tokens “million” and “currencysign” occur frequently due to the financial nature of the data.

Exhibit 32 Most Frequently Used Tokens in the Corpus



The most frequent word occurrences in the sentences in the negative sentiment and the positive sentiment classes are shown in Exhibit 33. The most commonly occurring words are similar for both sentiment classes, meaning that they are not useful in discriminating between the two sentiment classes. This finding demonstrates the utility of removing the most commonly used tokens from the BOW.

Exhibit 33 Most Frequently Used Tokens in Two Sentiment Classes of the Corpus

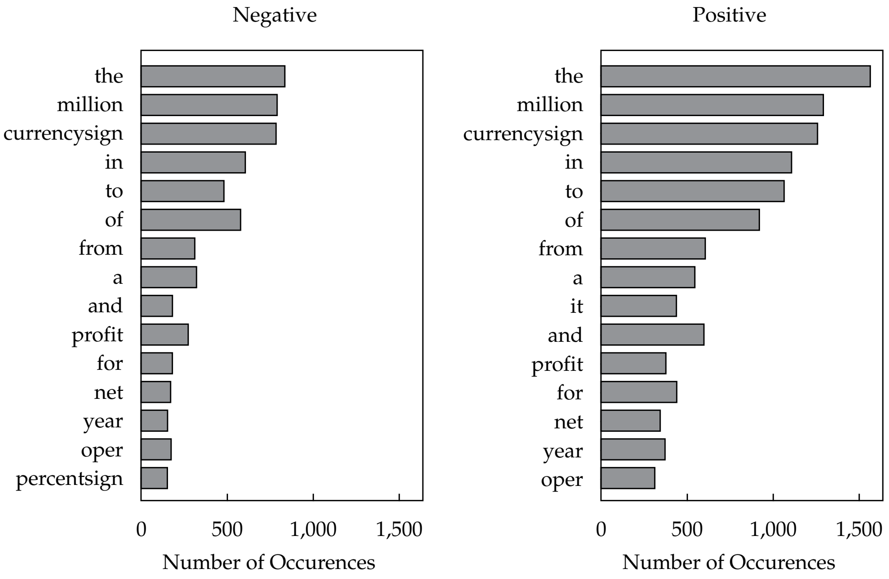
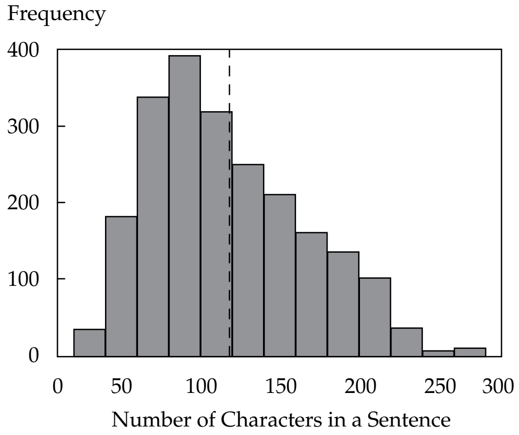


Exhibit 34 shows a histogram of sentence length distribution. **Sentence length** is defined as the number of characters, including spaces, in a sentence. The longest sentence has 273 characters; the shortest sentence has 26 characters; and the average number of characters is about 120 (indicated by the vertical line). Although this distribution does not have any direct impact on model training, this histogram visually demonstrates the range of sentence lengths and helps identify any extremely long or short sentences. This histogram does not appear unusual, so no outlier sentences need to be removed.

Exhibit 34 Histogram of Sentence Lengths with Mean Sentence Length



Word clouds are a convenient method of visualizing the text data because they enable rapid comprehension of a large number of tokens and their corresponding weights. Exhibit 35 shows a word cloud for all the sentences in the corpus. The font sizes of the words are proportionate to the number of occurrences of each word in the corpus. Similarly, Exhibit 36 shows the word cloud divided into two halves: one half representing negative sentiment class sentences (upper half); one half representing positive sentiment class sentences (lower half). Notably, some highly discriminative stems and words, such as "decreas" and "down" in the negative half and "increas" and "rose" in the positive half, are present. The feature selection process will eliminate common words and highlight useful words for better model training.

Exhibit 35 Word Cloud of Entire Corpus

in
currencysign
million
of
the
from
to
year
percentsign
profit
currencysign
million
sale
from
and
in
period
for

Exhibit 36 Word Cloud Divided by Two Sub-Groups of the Corpus

negative
quarter
total
loss
profit
decreas
down
million
fell
of
the
currencysign
increas
up
and
rose
positive

Feature Selection

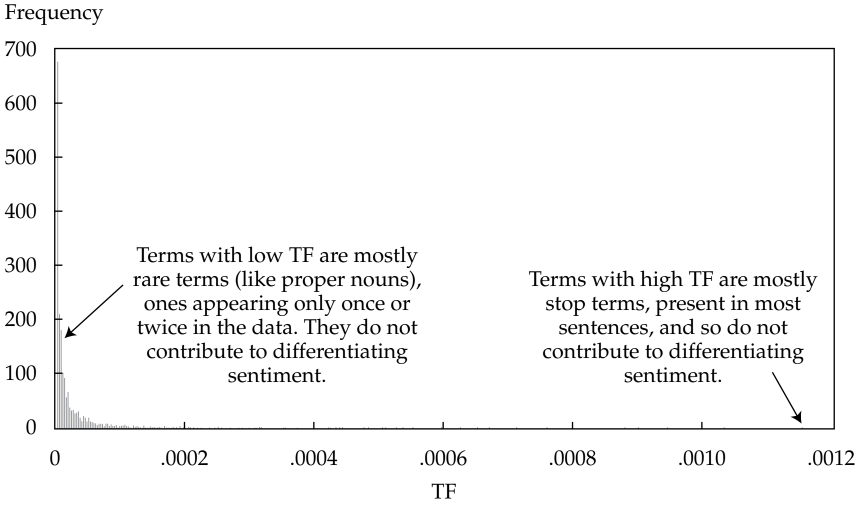
Exploratory data analysis revealed the most frequent tokens in the texts that could potentially add noise to this ML model training process. In addition to common tokens, many rarely occurring tokens, often proper nouns (i.e., names), are not informative for understanding the sentiment of the sentence. Further analyses must be conducted to decide which words to eliminate. Feature selection for text data involves keeping the useful tokens in the BOW that are informative and help to discriminate different classes of texts—those with positive sentiment and those with negative sentiment. At this point, a total of 44,151 non-unique tokens are in the 2,180 sentences.

Frequency analysis on the processed text data helps in filtering unnecessary tokens (or features) by quantifying how important tokens are in a sentence and in the corpus as a whole. Term frequency (TF) at the corpus level—also known as **collection frequency (CF)**—is the number of times a given word appears in the whole corpus (i.e., collection of sentences) divided by the total number of words in the corpus. Term frequency can be calculated and examined to identify outlier words. Exhibit 37 shows the descriptive statistics of term frequency for the words at the collection level. The statistics of TF range between 0 and 1 because TF values are ratios of total occurrences of a particular word to total number of words in the collection. A sample of words with the highest TF and lowest TF values is also shown to gain insight into what kinds of words occur at these extreme frequencies.

Exhibit 37 Summary Statistics of TF for Words at the Collection Level, Sample Words with High and Low TF Values, and Histogram of TF Values

```
    Min.    1st Qu.    Median     Mean    3rd Qu.     Max.
2.265e-05 2.265e-05 4.530e-05 3.741e-04 1.585e-04 5.429e-02

         word          TF              word            TF
        <chr>        <dbl>            <chr>          <dbl>
          the   0.05429096              yet   2.264954e-05
      million   0.04722430             yihn   2.264954e-05
 currencysign   0.04627302            young   2.264954e-05
           in   0.03870807         zahariev   2.264954e-05
           to   0.03476705             zone   2.264954e-05
           of   0.03377047              zoo   2.264954e-05
```



Calculating highest and lowest TFs at the collection level is a general strategy to identify noisy terms. The histogram in Exhibit 37 shows a long tail to the right, which represents common terms that must be removed. The high frequency bars on the left show that there are also many rare terms (e.g., ones appearing only once or twice across the data). Such rare terms do not appear enough to be used as meaningful features and are often removed. The words with the highest TF are mostly stop words that are not useful because they are present in most of the sentences and thus do not contribute to differentiating the sentiment embedded in the text. The words with the lowest TF values are mostly proper nouns or sparse terms that are also not important to the meaning of the text. In this example, after careful examination of words with extreme frequencies, the words with high TF values (>99.5th percentile, 14 words) and low TF values (<30th percentile, 714 words) are removed before forming the final document term matrix (DTM). Exhibit 38 shows the 14 words with the highest TF values (>99.5th percentile) that are the custom stop words for this project.

Exhibit 38 Fourteen Custom Stop Words for the Project

"the"	"million"	"currencysign"	"in"	"to"	"of"	"from"
"and"	"profit"	"for"	"it"	"not"	"year"	"a"

To construct a DTM for ML training, different TF measures need to be computed to fill in the cells of the DTM. Exhibit 39 displays part of a TF measures table that is computed for the text data before the removal of custom stop words.

Exhibit 39 Sample Output of Term Frequency (TF) Measures Table

SentenceNo	TotalWordsInSentence	Word	TotalWordCount	WordCountInSentence	SentenceCountWithWord	TF	DF	IDF	TFIDF
<int>	<int>	<chr>	<int>	<int>	<int>	<dbl>	<dbl>	<dbl>	<dbl>
624	34	a	873	6	687	0.1764706	0.3151376	1.1547459	0.20377868
701	39	the	2397	6	1453	0.1538462	0.6665138	0.4056945	0.06241454
1826	34	a	873	6	687	0.1764706	0.3151376	1.1547459	0.20377868
1963	39	the	2397	6	1453	0.1538462	0.6665138	0.4056945	0.06241454
128	30	of	1491	5	984	0.1666667	0.4513761	0.7954543	0.13257571
223	37	the	2397	5	1453	0.1351351	0.6665138	0.4056945	0.05482358

The columns of the term frequency measures table are as follows:

1 *SentenceNo*: A unique identification number assigned to each sentence in the order they are present in the original dataset. For example, sentence number 701 is a sentence in row 701 from the data table: "*the airlin estim that the cancel of it flight due to the closur of european airspac and the process of recommenc traffic have caus a the compani a loss of currencysign million includ the cost of strand passeng accommod*."

2 *TotalWordsInSentence*: Count of total number of words present in the sentence. For example, sentence number 701 has a total of 39 words.

3 *Word*: A word token that is present in the corresponding sentence.

4 *TotalWordCount*: Total number of occurrences of the word in the entire corpus or collection. For example, the token "the" occurs 2,397 times in the whole collection of sentences. The following equation can be used to compute TF at the collection level:

TF (Collection Level) =
TotalWordCount/Total number of words in collection. **(10)**

The TF of the word "the" at the collection level is calculated as 2,397/44,151 = 0.05429096. Note that this result was seen previously in Exhibit 37.

5 *WordCountInSentence*: Number of times the token is present in the corresponding sentence. For example, token "the" is present six times in sentence number 701.

6 *SentenceCountWithWord*: Number of sentences in which the word is present. For example, the token "the" is present in 1,453 sentences.

7 *TF (Term Frequency) at Sentence Level*: Number of times a word is present in a sentence divided by the total number of words in that sentence. The following equation can be used to compute TF at the sentence level:

TF (Sentence Level) =
WordCountInSentence/TotalWordsInSentence. **(11)**

For example, TF at the sentence level for the word "the" in sentences number 701 and 223 is calculated as 6/39 = 0.1538462 and 5/37 = 0.1351351, respectively.

8 *DF (Document Frequency)*: Defined as the number of documents (i.e., sentences) that contain a given word divided by the total number of sentences (here, 2,180). Document frequency is important since words frequently occurring across sentences provide no differentiating information in each sentence. The following equation can be used to compute DF:

$$\text{DF} = \text{SentenceCountWithWord/Total number of sentences.} \tag{12}$$

For example, DF of the word "the" is 1,453/2,180 = 0.6665138; so, 66.7% of the sentences contain the word "the." A high DF indicates high word frequency in the text.

9 *IDF (Inverse Document Frequency)*: A relative measure of how unique a term is across the entire corpus. Its meaning is not directly related to the size of the corpus. The following equation can be used to compute IDF:

$$\text{IDF} = \log(1/\text{DF}). \tag{13}$$

For example, IDF of the word "the" is log(1/0.6665138) = 0.4056945. A low IDF indicates high word frequency in the text.

10 *TF–IDF*: To get a complete representation of the value of each word, TF at the *sentence level* is multiplied by the IDF of a word across the entire dataset. Higher TF–IDF values indicate words that appear more frequently within a smaller number of documents. This signifies relatively more unique terms that are important. Conversely, a low TF–IDF value indicates terms that appear in many documents. TF–IDF values can be useful in measuring the key terms across a compilation of documents and can serve as word feature values for training an ML model. The following equation can be used to compute TF–IDF:

$$\text{TF–IDF} = \text{TF} \times \text{IDF}. \tag{14}$$

For example, TF–IDF of the token "of" is calculated as 0.1666667 × 0.7954543 = 0.13257571.

Similarly, Exhibit 40 shows high TF–IDF words for the text data before the removal of custom stop words.

Exhibit 40 Sample Output of High TF–IDF Words

```
SentenceNo TotalWordsInSentence     Word TotalWordCount WordCountInSentence SentenceCountWithWord        TF           DF      IDF     TFIDF
     <int>                <int>    <chr>          <int>               <int>                 <int>     <dbl>        <dbl>    <dbl>     <dbl>
        28                    7    risen              3                   1                     3 0.1428571 0.0013761468 6.588468 0.9412097
       830                    7 diminish              2                   1                     2 0.1428571 0.0009174312 6.993933 0.9991333
      1368                    9    great              4                   1                     4 0.1111111 0.0018348624 6.300786 0.7000873
      1848                    8    injuri             1                   1                     1 0.1250000 0.0004587156 7.687080 0.9608850
      1912                    7  cheaper              1                   1                     1 0.1428571 0.0004587156 7.687080 1.0981543
      1952                    6 argument              1                   1                     1 0.1666667 0.0004587156 7.687080 1.2811800
```

TF or TF–IDF values are placed at the intersection of sentences (rows) and terms (columns) of the document term matrix. For this project, TF values are used for the DTM as the texts are sentences rather than paragraphs or other larger bodies of text. TF–IDF values vary by the *number* of documents in the dataset; therefore, the model performance can vary when applied to a dataset with just a few documents. In addition to removing custom stop words and sparse terms, single character letters are also eliminated because they do not add any value to the sentiment significance.

Feature Engineering

N-grams are used as a feature engineering process in this project. Use of n-grams helps to understand the sentiment of a sentence as a whole. As mentioned previously, the objective of this project is to predict sentiment class (positive and negative) from financial texts. Both unigram and bigrams are implemented, and the BOW is created from them. Bigram tokens are helpful for keeping negations intact in the text, which is vital for sentiment prediction. For example, the tokens "not" and "good" or "no" and "longer" can be formed into single tokens, now bigrams, such as "not_good" and "no_longer." These and similar tokens can be useful during ML model training and can improve model performance. Exhibit 41 shows a sample of 100 words from the BOW containing both unigram and bigram tokens after removal of custom stop words, sparse terms, and single characters. Note that the BOW contains such tokens as increas, loss, loss_prior, oper_rose, tax_loss, and sale_increas. Such tokens are informative about the embedded sentiment in the texts and are useful for training an ML model. The corresponding word frequency measures for the document term matrix are computed based on this new BOW.

Exhibit 41 One-Hundred Sample Tokens from Final BOW of Entire Corpus

"last"	"last_quarter"	"quarter"	"quarter_componenta"	"componenta"
"componenta_sale"	"sale"	"sale_doubl"	"doubl"	"doubl_same"
"same"	"same_period"	"period"	"period_earlier"	"earlier"
"earlier_while"	"while"	"while_move"	"move"	"move_zero"
"zero"	"zero_pre"	"pre"	"pre_tax"	"tax"
"tax_pre"	"tax_loss"	"loss"	"third"	"third_quarter"
"quarter_sale"	"sale_increas"	"increas"	"increas_by"	"by"
"by_percentsign"	"percentsign"	"percentsign_oper"	"oper"	"oper_by"
"oper_rose"	"rose"	"rose_correspond"	"correspond"	"correspond_period"
"period_repres"	"repres"	"repres_percentsign"	"percentsign_sale"	"oper_total"
"total"	"total_up"	"up"	"up_repres"	"finnish"
"finnish_talentum"	"talentum"	"talentum_report"	"report"	"report_oper"
"oper_increas"	"increas_sale"	"sale_total"	"cloth"	"cloth_retail"
"retail"	"retail_chain"	"chain"	"chain_sepp"	"sepp"
"sepp_ls"	"ls"	"ls_sale"	"consolid"	"consolid_sale"
"incres_percentsign"	"percentsign_reach"	"reach"	"reach_while"	"while_oper"
"oper_amount"	"amount"	"amount_compar"	"compar"	"compar_loss"
"loss_prior"	"prior"	"prior_period"	"foundri"	"foundri_divis"
"divis"	"divis_report"	"report_sale"	"percentsign_correspond"	"period_sale"
"sale_machin"	"machin"	"machin_shop"	"shop"	"shop_divis"

EXAMPLE 6

Calculating and Interpreting Term Frequency Measures

Data scientists Jack and Jill are using financial text data to develop sentiment indicators for forecasting future stock price movements. They have assembled a BOW from the corpus of text being examined and have pulled the following abbreviated term frequency measures tables.

Term Frequency Measures Table 1

SentenceNo	TotalWordsInSentence	Word	TotalWordCount	WordCountInSentence	SentenceCountWithWord
<int>	<int>	<chr>	<int>	<int>	<int>
624	34	a	873	6	687
701	39	the	2397	6	1453
1826	34	a	873	6	687
1963	39	the	2397	6	1453
128	30	of	1491	5	984
223	37	the	2397	5	1453

Term Frequency Measures Table 2

SentenceNo	TotalWordsInSentence	Word	TotalWordCount	WordCountInSentence	SentenceCountWithWord
<int>	<int>	<chr>	<int>	<int>	<int>
28	7	risen	3	1	3
830	7	diminish	2	1	2
1368	9	great	4	1	4
1848	8	injuri	1	1	1
1912	7	cheaper	1	1	1
1952	6	argument	1	1	1

1 Determine and interpret term frequency (TF) at the collection level and at the sentence level for the word (i.e., token) "a" in sentence 1,826 in term frequency measures Table 1 and then for the token "great" in sentence 1,368 in term frequency measures Table 2.

2 Determine and interpret TF–IDF (term frequency–inverse document frequency) for the word "a" in sentence 1,826 in term frequency measures Table 1 and then for the token "great" in sentence 1,368 in term frequency measures Table 2.

Solution to 1:

TF at the collection level is calculated using Equation 10:

TF (Collection Level) = TotalWordCount/Total number of words in collection.

For token "a" in sentence 1,826 (Table 1), TF (Collection Level) is 873/44,151 = 0.019773 or 1.977%.For token "great" in sentence 1,368 (Table 2), TF (Collection Level) is 4/44,151 = 0.000091 or 0.009%.TF at the collection level is an indicator of the frequency, in percentage terms, that a token is used throughout the whole collection of texts (here, 44,151). It is useful for identifying outlier words: Tokens with highest TF values are mostly stop words that do not contribute to differentiating the sentiment embedded in the text (such as "a"), and tokens with lowest TF values are mostly proper nouns or sparse terms that are also not important to the meaning of the text. Conversely, tokens with intermediate TF values potentially carry important information useful for differentiating the sentiment embedded in the text.TF at the sentence level is calculated using Equation 11:

TF (Sentence Level) = WordCountInSentence/TotalWordsInSentence.

For token "a" in sentence 1,826, TF (Sentence Level) is 6/34 = 0.176471 or 17.647%.

For token "great" in sentence 1,368, TF (Sentence Level) is 1/9 = 0.111111 or 11.111%.

TF at the sentence level is an indicator of the frequency, in percentage terms, that a token is used in a particular sentence (i.e., instance). Therefore, it is useful for understanding the importance of the specific token in a given sentence.

Solution to 2:

To calculate TF–IDF, besides TF at the sentence level, document frequency (DF) and inverse document frequency (IDF) are also required.

DF is the number of documents (i.e., sentences) that contain a given word divided by the total number of sentences in the corpus (here, 2,180). DF is calculated using Equation 12:

DF = SentenceCountWithWord/Total number of sentences.

For token “a” in sentence 1,826, DF is 687/2,180 = 0.315138 or 31.514%.

For token “great” in sentence 1,368, DF is 4/2,180 = 0.001835 or 0.184%.

Document frequency is important since tokens occurring frequently across sentences (such as “a”) provide no differentiating information in each sentence. Tokens occurring less frequently across sentences (such as “great”), however, may provide useful differentiating information.

IDF is a relative measure of how important a term is across the entire corpus (i.e., collection of texts/sentences). IDF is calculated using Equation 13:

IDF = log(1/DF).

For token “a” in sentence 1,826, IDF is log(1/0.315138) = 1.154746.

For token “great” in sentence 1,368, IDF is log(1/0.001835) = 6.300786.

Using TF and IDF, TF–IDF can now be calculated using Equation 14:

TF–IDF = TF × IDF.

For token “a” in sentence 1,826, TF–IDF = 0.176471 × 1.154746 = 0.203779, or 20.378%.

For token “great” in sentence 1,368, TF–IDF = 0.111111 × 6.300786 = 0.700087, or 70.009%.

As TF–IDF combines TF at the *sentence level* with IDF across the entire corpus, it provides a complete representation of the value of each word. A high TF–IDF value indicates the word appears many times within a small number of documents, signifying an important yet unique term within a sentence (such as “great”). A low TF–IDF value indicates tokens that appear in most of the sentences and are not discriminative (such as “a”). TF–IDF values are useful in extracting the key terms in a document for use as features for training an ML model.

7.3 Model Training

The sentiment class labels (positive and negative) constitute the target variable (*y*) for model training. They are relabeled as 1 (for positive) and 0 (for negative) to enable calculating the performance metrics, such as receiver operating characteristic (ROC) curve and area under the curve (AUC) from the trained model results. The master dataset that has been cleansed and preprocessed is partitioned into three separate sets: 1) training set; 2) cross-validation (CV) set; and 3) test set. These are in the ratio of 60:20:20, respectively (following common practice). For splitting, simple random sampling is applied within levels of the target variable to balance the class distributions within the splits. The final DTM is built using the sentences (rows), which are the instances, and resulting tokens (columns), which are the feature variables, from the BOW of the training dataset. The final BOW consists of unigram and bigram tokens from the sentences in the training corpus only. The DTM is then filled in with resultant TF values of the tokens from the training corpus.

Similarly, the DTMs for the CV set and the test set are built using tokens from the final training BOW for tuning, validating, and testing of the model. To be clear, the final BOW from the training corpus is used for building DTMs across all the splits because the model has been trained on that final BOW. Thus, the columns (think, features) of all three DTMs are the same, but the number of rows varies because a different number of sentences are in each split. The DTMs are filled with resultant term frequency values calculated using sentences in the corpuses of the respective splits—sentences from the CV set corpus and sentences from the test set corpus. Exhibit 42 tabulates the summary of dimensions of the data splits and their uses in the model training process. As mentioned, the columns of DTMs for the splits are the same, equal to the number of unique tokens (i.e., features) from the final training corpus BOW, which is 9,188. Note that this number of unique tokens (9,188) differs from that in the master corpus (11,501) based on the sentences that are included in the training corpus after the random sampling.

Exhibit 42 Summary of the Three Data Splits

Corpus	Split %	Number of Sentences	DTM Dimensions	Purpose
Master	100%	2180	2180 × 11501	Used for data exploration
Training	60%	1309	1309 × 9188	Used for ML model training
CV	20%	435	435 × 9188	Used for tuning and validating the trained model
Test	20%	436	436 × 9188	Used for testing the trained, tuned, and validated model

Method Selection

Alternative ML methods, including SVM, decision trees, and logistic regression, were examined because these techniques are all considered potentially suitable for this particular task (i.e., supervised learning), type of data (i.e., text), and size of data (i.e., wider data with many potential variables). The SVM and logistic regression methods appeared to offer better performance than decision trees. For brevity, we discuss logistic regression in the remainder of the chapter. Logistic regression was used to train the model, using the training corpus DTM containing 1,309 sentences. As a reminder, in this project texts are the sentences and the classifications are positive and negative sentiment classes (labeled 1 and 0, respectively). The tokens are feature variables, and the sentiment class is the target variable. Text data typically contain thousands of tokens. These result in sparse DTMs because each column represents a token feature and the values are mostly zeros (i.e., not all the tokens are present in every text). Logistic regression can deal with such sparse training data because the regression coefficients will be close to zero for tokens that are not present in a significant number of sentences. This allows the model to ignore a large number of minimally useful features. Regularization further helps lower the coefficients when the features rarely occur and do not contribute to the model training.

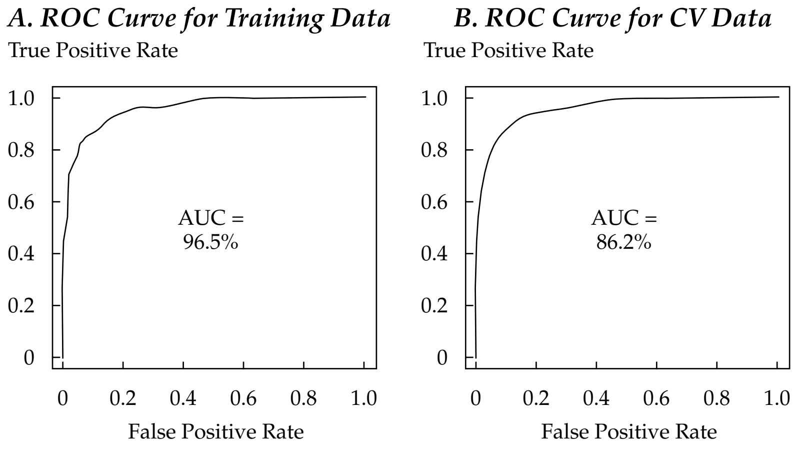
Logistic regression is applied on the final training DTM for model training. As this method uses maximum likelihood estimation, the output of the logistic model is a probability value ranging from 0 to 1. However, because the target variable is binary, coefficients from the logistic regression model are not directly used to predict the value of the target variable. Rather, a mathematical function uses the logistic regression coefficient (β) to calculate probability (p) of sentences having positive sentiment (y =

1).[3] If p for a sentence is 0.90, there is a 90% likelihood that the sentence has positive sentiment. Theoretically, the sentences with p > 0.50 likely have positive sentiment. Because this is not always true in practice, however, it is important to find an ideal threshold value of p. We elaborate on this point in a subsequent example. The threshold value is a cutoff point for p values, and the ideal threshold p value is influenced by the dataset and model training. When the p values (i.e., probability of sentences having positive sentiment) of sentences are above this ideal threshold p value, then the sentences are *highly* likely to have positive sentiment (y = 1). The ideal threshold p value is estimated heuristically using performance metrics and ROC curves, as will be demonstrated shortly.

Performance Evaluation and Tuning

The trained ML model is used to predict the sentiments of the sentences in the training and CV DTMs. Exhibit 43 displays the ROC curves for the training (Panel A) and CV (Panel B) data. Remember that the x-axis is false positive rate, FP/(TN + FP), and the y-axis is true positive rate, TP/(TP + FN). As the model is trained using the training DTM, it clearly performs well on the same training data (so there is no concern about underfitting) but does not perform as well on the CV data. This is apparent as the ROC curves are significantly different between the training and CV datasets. The AUC is 96.5% on training data and 86.2% on CV data. This finding suggests that the model performs comparatively poorly (with a higher rate of error or misclassification) on the CV data when compared to training data. Thus, the implication is that the model is overfitted.

Exhibit 43 ROC Curves of Model Results for Training and CV Data Before Regularization



A. ROC Curve for Training Data

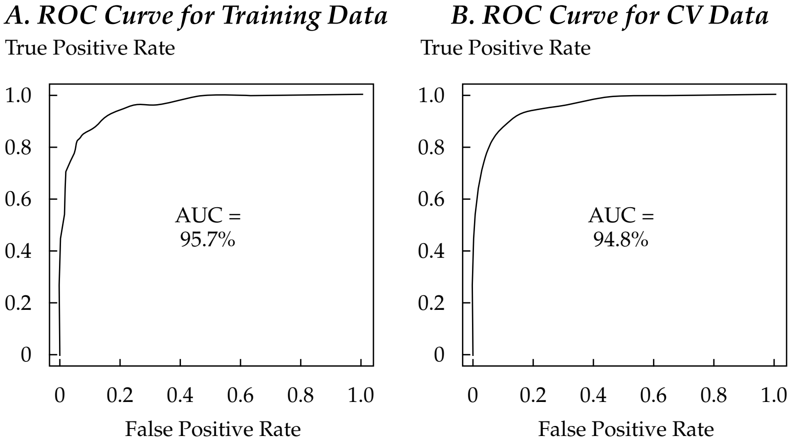
B. ROC Curve for CV Data

As the model is overfitted, least absolute shrinkage and selection operator (LASSO) regularization is applied to the logistic regression. LASSO regularization penalizes the coefficients of the logistic regression to prevent overfitting of the model. The penalized regression will select the tokens (features) that have statistically significant (i.e., non-zero) coefficients and that contribute to the model fit; LASSO does this while disregarding the other tokens. Exhibit 44 shows the ROC curves for the new model

[3] This mathematical function is an exponential function of the form: $P\,(y = 1) = \dfrac{1}{1 + \exp^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n)}}$ where the βs are the logistic regression coefficients.

that uses regularized logistic regression. The ROC curves look similar for model performance on both datasets, with an AUC of 95.7% on the training dataset (Panel A) and 94.8% on the CV dataset (Panel B). These findings suggest that the model performs similarly on both training and CV data and thus indicate a good fitting model (one that is not overfitted).

Exhibit 44 ROC Curves of Model Results for Training and CV Data After Regularization


A. ROC Curve for Training Data

B. ROC Curve for CV Data

Regularization along with careful feature selection help to prevent overfitting in logistic regression models. Another model was trained using all token features, including stop words, sparse terms, and single characters, with no regularization. That model showed an AUC of 99.1% when applied on the training dataset and an AUC of 89.4% when applied on the CV dataset, suggesting that the model is overfitting. As the AUC values in all of the models discussed are not far from 100%, these models are clearly not underfitting. In sum, the final ML model for this project uses logistic regression with LASSO regularization.

To further evaluate the model, error analysis is conducted by calculating a confusion matrix using the ML model results from the cross-validation dataset. The threshold p value of 0.5 is used as a cutoff point. When target value p > 0.5, the prediction is assumed to be $y = 1$ (meaning, positive sentiment). Otherwise, the prediction is assumed to be $y = 0$ (negative sentiment). A confusion matrix, with performance metrics and overall scores for the model results using the CV data, is shown in Exhibit 45.

Exhibit 45 **Confusion Matrix of Model Results for CV Data with Threshold p Value = 0.50**

Confusion Matrix for CV Data with Threshold = 0.5

		Actual Training Labels	
		Class "1"	Class "0"
Predicted Results	Class "1"	284 (TP)	38 (FP)
	Class "0"	7 (FN)	106 (TN)

Performance Metrics

$TP = 284, FP = 38, FN = 7, TN = 106$

$P = 284 / (284+38) = 0.88$

$R = 284 / (284+7) = 0.98$

$\text{F1 Score} = (2 \times 0.88 \times 0.98) / (0.88 + 0.98) = 0.93$

$\text{Accuracy} = (284 + 106) / (284 + 38 + 106 + 7) = 0.90$

The model accuracy is 90% with a theoretically suggested (default) threshold p value of 0.5. The CV data are used to tune the threshold value for best model performance. Various p values from 0.01 to 0.99 are systematically evaluated individually, and confusion matrixes and performance metrics are calculated using each of these p values. Based on these metrics, the p value resulting in the highest model accuracy is selected as the ideal threshold p value. However, there are often trade-offs: Minimizing false positives (FPs) comes at a cost of increasing false negatives (FNs), and vice versa. Prioritizing various performance statistics (e.g., precision versus recall) depends on the context and relative consequences of FP and FN on the project applications. In this project, the values of negative sentiment and positive sentiment sentences are assumed to be equal, thus the impacts of FP and FN are also equal. It is common practice to simulate many model results using different threshold p values and to search for maximized accuracy and F1 statistics that minimize these trade-offs. As noted earlier, accuracy and F1 scores are overall performance measures that give equal weight to FP and FN.

Exhibit 46 shows the overall performance measures (i.e., F1 score and accuracy) for various threshold p values. The threshold p value that results in the highest accuracy and F1 score can now be identified. From the charts in Exhibit 45, the ideal threshold p value appears to be around 0.60. To investigate further, a table of performance measures (i.e., precision, recall, F1 score, and accuracy) is generated for a series of threshold p values ranging from 0.45 to 0.75. The table in Exhibit 47 demonstrates that threshold p values between 0.60 and 0.63 result in the highest accuracy and F1 score for the CV dataset. As a result of this analysis, a final threshold p value of 0.60 is selected.

Exhibit 46 Threshold Values Versus Overall Performance Measures

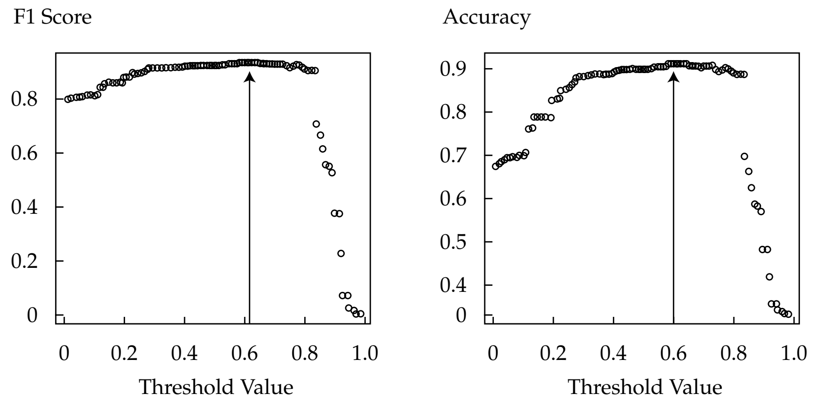


Exhibit 47 Performance Measures of the Model for a Series of Threshold Values

Threshold	Precision	Recall	F1	Accuracy
0.45	0.8750000	0.986254296	0.927302100	0.8965517
0.46	0.8827160	0.982817869	0.930081301	0.9011494
0.47	0.8827160	0.982817869	0.930081301	0.9011494
0.48	0.8819876	0.975945017	0.926590538	0.8965517
0.49	0.8819876	0.975945017	0.926590538	0.8965517
0.50	0.8819876	0.975945017	0.926590538	0.8965517
0.51	0.8819876	0.975945017	0.926590538	0.8965517
0.52	0.8819876	0.975945017	0.926590538	0.8965517
0.53	0.8902821	0.975945017	0.931147541	0.9034483
0.54	0.8930818	0.975945017	0.932676519	0.9057471
0.55	0.8930818	0.975945017	0.932676519	0.9057471
0.56	0.8958991	0.975945017	0.934210526	0.9080460
0.57	0.8958991	0.975945017	0.934210526	0.9080460
0.58	0.8958991	9.975945017	0.934210526	0.9080460
0.59	0.9015873	0.975945017	0.937293729	0.9126437
0.60	0.9044586	0.975945017	0.938842975	0.9149425
0.61	0.9044586	0.975945017	0.938842975	0.9149425
0.62	0.9044586	0.975945017	0.938842975	0.9149425
0.63	0.9041534	0.972508591	0.937086093	0.9126437
0.64	0.9041534	0.972508591	0.937086093	0.9126537
0.65	0.9041534	0.972508591	0.937086093	0.9126437
0.66	0.9035370	0.965635739	0.933554817	0.9080460
0.67	0.9035370	0.965635739	0.933554817	0.9080460
0.68	0.9064516	0.965635739	0.935108153	0.9103448
0.69	0.9064516	0.965635739	0.935108153	0.9103448
0.70	0.9061489	0.962199313	0.933333333	0.9080460
0.71	0.9061489	0.962199313	0.933333333	0.9080460

(continued)

Exhibit 47 (Continued)

Threshold	Precision	Recall	F1	Accuracy
0.72	0.9090909	0.962199313	0.934891486	0.9103448
0.73	0.9090909	0.962199313	0.934891486	0.9103448
0.74	0.9078947	0.948453608	0.927731092	0.9011494
0.75	0.9072848	0.941580756	0.924114671	0.8965517

* The shaded row shows the selected threshold p value (0.60) and the performance metrics for the selected model.

Finally, the confusion matrix using the ideal threshold p value of 0.60 is constructed to observe the performance of the final model. When target value p > 0.60, the prediction is assumed to be $y = 1$ (indicating positive sentiment); otherwise, the prediction is assumed to be $y = 0$ (negative sentiment). The confusion matrix for the CV data is shown in Exhibit 48. It is clear that the model performance metrics have improved in the final model compared to the earliest case when the threshold p value was 0.50. Now, accuracy and F1 score have both increased by one percentage point to 91% and 94%, respectively, while precision has increased by two percentage points to 90%.

Exhibit 48 Confusion Matrix of Model Results for CV Data with Threshold p Value = 0.60

Confusion Matrix for CV Data with Threshold = 0.6

		Actual Training Labels	
		Class "1"	Class "0"
Predicted Results	Class "1"	284 (TP)	30 (FP)
	Class "0"	7 (FN)	114 (TN)

Performance Metrics

TP = 284, FP = 30, FN = 7, TN = 114

P = 284 / (284+30) = 0.90

R = 284 / (284+7) = 0.98

F1 Score = (2 × 0.90 × 0.98) / (0.90 + 0.98) = 0.94

Accuracy = (284 + 114) / (284 + 30 + 114 + 7) = 0.91

7.4 Results and Interpretation

The final ML model with the appropriate threshold p value has been validated and is now ready for use. The model can be used to predict the sentiment of new sentences from the test data corpus as well as new sentences from similar financial text data sources, such as news wires, earnings call transcripts, and quarterly financial reports. The final model is a collection of penalized regression coefficients for unigram and bigram tokens from the BOW of the training corpus. To use the model to predict the sentiment of new sentences, tokenization and identical cleansing and preprocessing operations must be performed on the new sentences. All the processes performed on the training data must be performed on the new data to which the model will be applied (as was done for the test dataset). The model will use the trained penalized regression coefficients on the term frequency (TF) values of the tokens in the document term matrix (DTM) of the new sentences and will determine the target value (p). The columns of the DTM of the new sentences are the same as those of the training DTM, but the TF values are calculated based on the test corpus. Using the threshold p value of 0.60, the sentiment class for each sentence in the test corpus will be predicted.

The model is now applied on the test data that contains 436 sentences. Note that the test data were not used to train or validate/tune the model and are new to the model. The test data were preprocessed identically to the training and CV data while a part of the master corpus. The model is then applied to the test DTM, and the results are obtained. Exhibit 49 displays 30 sample results from the test corpus. The results table contains cleansed and preprocessed sentences, actual sentiment, target p values from the model, and predicted sentiment. Note that this sample contains three cases of misclassification: the 10th sentence (text), where p = 0.46; the 26th text, where p = 0.77; and the 30th text, where p = 0.71. Therefore, accuracy of this 30-text sample is 27/30 = 90%.

Exhibit 49 Thirty Sample Results of Test Data

Sentence	Sentiment	p	Predicted Sentiment
exclude non recur item pre tax surg percentsign	1	0.81	1
adp news feb finnish retail kesko oyj hel kesbv said today total sale exclud valu ad tax vat stood at januari down percentsign on yea	0	0.12	0
india trade with russia current stand at four billion dollar grow per cent fiscal	1	0.83	1
refin margin was bbl combar bbl prior	1	0.81	1
scania morgan Stanley lift share target on swedish heavi duti truck bus maker scania ab crown euro crown euro	1	0.83	1
deal is like bring save	1	0.83	1
will also strengthen ruukki offshore busi	1	0.83	1
last week finnish metl technolog group announc plan sell more than percent technolog unit further compani strategy goal becom world largest stainless steel maker	1	0.83	1
nest oil board propos dividend full compar with ago	1	0.81	1
pre tax loss total compar loss first quarter	1	0.46	0
pretax total compar loss fourth quarter	1	0.74	1
re use back into pet bottle has also steadili increas rate use strap tape has pick up again after dip pector said previous	1	0.95	1
satama sale would be higher than befor	1	0.83	1
octob finnish wood product technolog supplier raut oyj hel rutav said today swung first nine month versus loss same period earlier	1	0.79	1
ebit total compar loss correspond period	1	0.74	1
finnish consum packag manufactur huhtamaki oyj said swung euro first nine month loss euro same period	1	0.77	1
finnish dental care group oral hammaslaakarit oyj post total euro first nine month versus loss euro same period	1	0.79	1
finnish silicon water manufactur okmet oyj said swung euro first nine month loss euro earlier	1	0.77	1
adp news feb finnish print circuit board pcb maker aspocomp group oyj hel acg said today swung versus loss	1	0.79	1
mn pretax third quarter	1	0.83	1

(continued)

Exhibit 49 (Continued)

Sentence	Sentiment	p	Predicted Sentiment
oper total compar correspond period	1	0.81	1
raut post euro third quarter compar loss euro correspond period	1	0.74	1
russian export duti will active harvest finland sale russia will increas also	1	0.91	1
compani expect sale signific increas	1	0.91	1
compani amount ee which was percentsign more than	1	0.81	1
third quarter fiscal efor swung loss versus correspond period fiscal	0	0.77	1
acando ab acanb ss fell percent kronor lowest close sinc dec	0	0.20	0
compani oper loss total compar	0	0.27	0
last paseng flew airlin down percent	0	0.12	0
loss after financi item total compar correspond period	0	0.71	1

Exhibit 50 shows the confusion matrix for the test data. Accuracy and F1 score are 90% and 93%, respectively, while precision and recall are 89% and 98%, respectively. Therefore, it is apparent that the model performs similarly on the training, CV, and test datasets. These findings suggest that the model is robust and is not overfitting. They also suggest that the model should generalize well out-of-sample and can thus be used to predict the sentiment classes for new sentences from similar financial text data sources. Of course, these new text data must first be subjected to identical tokenization, cleansing, and preprocessing as done for the training dataset.

Exhibit 50 Confusion Matrix of Model Results for Test Data with Threshold p Value = 0.60

Confusion Matrix for Test Data

		Actual Training Labels	
		Class “1”	Class “0”
Predicted Results	Class “1”	284 (TP)	35 (FP)
	Class “0”	7 (FN)	110 (TN)

Performance Metrics

TP = 284, FP = 35, FN = 7, TN = 110

P = 284 / (284+35) = 0.89

R = 284 / (284+7) = 0.98

F1 Score = (2 × 0.89 × 0.98) / (0.89 + 0.98) = 0.93

Accuracy = (284 + 110) / (284 + 35 + 110 + 7) = 0.90

To recap, this project involves converting unstructured data (i.e., text data from financial data sources) into structured data (i.e., tokens, sentences, and term frequency values) in a document term matrix that is used as input for training, validating, and testing machine learning-based models (here, logistic regression) for predicting classification (here, sentiment classes). Similar models can be built and used in different contexts to understand the sentiment embedded in larger texts. The derived sentiment classification can be useful as a visualization tool to provide insight about the text without reading large documents. These sentiment classifications can also be used as structured input data for larger ML models that have a specific purpose, such as to predict future stock price movements.

EXAMPLE 7

Comparing Performance Metrics for Confusion Matrixes with Different Threshold p Values

In the previous analysis using the cross-validation dataset, performance measures for the sentiment classification ML model were calculated for a wide range (from 0.45 to 0.75) of threshold p values. The threshold value of 0.60 was determined to be the p value that maximizes model accuracy and F1 score; the confusion matrix for this model is shown in Exhibit 48. Use the following confusion matrixes with threshold p values of 0.75 and 0.45, A and B, respectively, to answer the following questions.

Confusion Matrix A

Confusion Matrix for CV, Threshold = 0.75

N = 436		Actual Training Labels	
		Class "1"	Class "0"
Predicted Results	Class "1"	281	28
	Class "0"	17	110

Performance Metrics

TP = 281, FP = 28, FN = 17, TN = 110
Precision = TP/(TP + FP) = 0.91
Recall = TP/(TP + FN) = 0.94
F1 Score = HMean: Prec. & Recall = 0.93
Accuracy = (TP + TN)/N = 0.90

Confusion Matrix B

Confusion Matrix for CV, Threshold = 0.45

N = 436		Actual Training Labels	
		Class "1"	Class "0"
Predicted Results	Class "1"	281	41
	Class "0"	4	110

Performance Metrics

TP = 281, FP = 41, FN = 4, TN = 110
Precision = TP/(TP + FP) = 0.87
Recall = TP/(TP + FN) = 0.99
F1 Score = HMean: Prec. & Recall = 0.93
Accuracy = (TP + TN)/N = 0.90

1 Compare the performance metrics of confusion matrix A (using a threshold p value of 0.75) with the confusion matrix in Exhibit 48 (using a threshold p value of 0.60).

2 Compare the performance metrics of confusion matrix B (using a threshold p value of 0.45) with the confusion matrix in Exhibit 48 (using a threshold p value of 0.60).

3 Contrast the performance metrics of confusion matrixes A and B, and explain the trade-offs implied between them.

Solution to 1:

Since confusion matrix A has fewer true positives (TPs) and fewer true negatives (TNs) than the confusion matrix in Exhibit 48 (281 vs. 284 and 110 vs. 114, respectively), confusion matrix A has lower accuracy and a lower F1 score compared to the one in Exhibit 48 (0.90 vs. 0.91 and 0.93 vs. 0.94, respectively). Also, although confusion matrix A has slightly better precision, 0.91 vs. 0.90, due to a few less false positives (FPs), it has significantly lower recall, 0.94 vs. 0.98, due to having many more false negatives (FNs), 17 vs. 7, than the confusion matrix in Exhibit 48. On balance, the ML model using the threshold p value of 0.60 is the superior model for this sentiment classification problem.

Solution to 2:

Confusion matrix B has the same number of TPs (281) and TNs (110) as confusion matrix A. Therefore, confusion matrix B also has lower accuracy (0.90) and a lower F1 score (0.93) compared to the one in Exhibit 48. Although confusion matrix B has slightly better recall, 0.99 vs. 0.98, due to fewer FNs, it has somewhat lower precision, 0.87 vs. 0.90, due to having many more FPs, 41 vs.

30, than the confusion matrix in Exhibit 48. Again, it is apparent that the ML model using the threshold p value of 0.60 is the better model in this sentiment classification context.

Solution to 3:

The main differences in performance metrics between confusion matrixes A and B are in precision and recall. Confusion matrix A has higher precision, at 0.91 vs. 0.87, but confusion matrix B has higher recall, at 0.99 vs. 0.94. These differences highlight the trade-off between FP (Type I error) and FN (Type II error). Precision is useful when the cost of FP is high, such as when an expensive product that is fine mistakenly fails quality inspection and is scrapped; in this case, FP should be minimized. Recall is useful when the cost of FN is high, such as when an expensive product is defective but mistakenly passes quality inspection and is sent to the customer; in this case, FN should be minimized. In the context of sentiment classification, FP might result in buying a stock for which sentiment is incorrectly classified as positive when it is actually negative. Conversely, FN might result in avoiding (or even shorting) a stock for which the sentiment is incorrectly classified as negative when it is actually positive. The model behind the confusion matrix in Exhibit 48 strikes a balance in the trade-off between precision and recall.

SUMMARY

In this reading, we have discussed the major steps in big data projects involving the development of machine learning (ML) models—namely, those combining textual big data with structured inputs.

- Big data—defined as data with volume, velocity, variety, and potentially lower veracity—has tremendous potential for various fintech applications, including several related to investment management.
- The main steps for traditional ML model building are conceptualization of the problem, data collection, data preparation and wrangling, data exploration, and model training.
- For textual ML model building, the first four steps differ somewhat from those used in the traditional model: Text problem formulation, text curation, text preparation and wrangling, and text exploration are typically necessary.
- For structured data, data preparation and wrangling entail data cleansing and data preprocessing. Data cleansing typically involves resolving incompleteness errors, invalidity errors, inaccuracy errors, inconsistency errors, non-uniformity errors, and duplication errors.
- Preprocessing for structured data typically involves performing the following transformations: extraction, aggregation, filtration, selection, and conversion.
- Preparation and wrangling text (unstructured) data involves a set of text-specific cleansing and preprocessing tasks. Text cleansing typically involves removing the following: html tags, punctuations, most numbers, and white spaces.

- Text preprocessing requires performing normalization that involves the following: lowercasing, removing stop words, stemming, lemmatization, creating bag-of-words (BOW) and n-grams, and organizing the BOW and n-grams into a document term matrix (DTM).
- Data exploration encompasses exploratory data analysis, feature selection, and feature engineering. Whereas histograms, box plots, and scatterplots are common techniques for exploring structured data, word clouds are an effective way to gain a high-level picture of the composition of textual content. These visualization tools help share knowledge among the team (business subject matter experts, quants, technologists, etc.) to help derive optimal solutions.
- Feature selection methods used for text data include term frequency, document frequency, chi-square test, and a mutual information measure. Feature engineering for text data includes converting numbers into tokens, creating n-grams, and using name entity recognition and parts of speech to engineer new feature variables.
- The model training steps (method selection, performance evaluation, and model tuning) often do not differ much for structured versus unstructured data projects.
- Model selection is governed by the following factors: whether the data project involves labeled data (supervised learning) or unlabeled data (unsupervised learning); the type of data (numerical, continuous, or categorical; text data; image data; speech data; etc.); and the size of the dataset.
- Model performance evaluation involves error analysis using confusion matrixes, determining receiver operating characteristics, and calculating root mean square error.
- To carry out an error analysis for each model, a confusion matrix is created; true positives (TPs), true negatives (TNs), false positives (FPs), and false negatives (FNs) are determined. Then, the following performance metrics are calculated: accuracy, F1 score, precision, and recall. The higher the accuracy and F1 score, the better the model performance.
- To carry out receiver operating characteristic (ROC) analysis, ROC curves and area under the curve (AUC) of various models are calculated and compared. The more convex the ROC curve and the higher the AUC, the better the model performance.
- Model tuning involves managing the trade-off between model bias error, associated with underfitting, and model variance error, associated with overfitting. A fitting curve of in-sample (training sample) error and out-of-sample (cross-validation sample) error on the y-axis versus model complexity on the x-axis is useful for managing the bias vs. variance error trade-off.
- In a real-world big data project involving text data analysis for classifying and predicting sentiment of financial text for particular stocks, the text data are transformed into structured data for populating the DTM, which is then used as the input for the ML algorithm.
- To derive term frequency (TF) at the sentence level and TF–IDF, both of which can be inputs to the DTM, the following frequency measures should be used to create a term frequency measures table: TotalWordsInSentence; TotalWordCount; TermFrequency (Collection Level); WordCountInSentence; SentenceCountWithWord; Document Frequency; and Inverse Document Frequency.

PRACTICE PROBLEMS

The following information relates to Questions 1–15

Aaliyah Schultz is a fixed-income portfolio manager at Aries Investments. Schultz supervises Ameris Steele, a junior analyst.

A few years ago, Schultz developed a proprietary machine learning (ML) model that aims to predict downgrades of publicly-traded firms by bond rating agencies. The model currently relies only on structured financial data collected from different sources. Schultz thinks the model's predictive power may be improved by incorporating sentiment data derived from textual analysis of news articles and Twitter content relating to the subject companies.

Schultz and Steele meet to discuss plans for incorporating the sentiment data into the model. They discuss the differences in the steps between building ML models that use traditional structured data and building ML models that use textual big data. Steele tells Schultz:

Statement 1 The second step in building text-based ML models is text preparation and wrangling, whereas the second step in building ML models using structured data is data collection.

Statement 2 The fourth step in building both types of models encompasses data/text exploration.

Steele expresses concern about using Twitter content in the model, noting that research suggests that as much as 10%–15% of social media content is from fake accounts. Schultz tells Steele that she understands her concern but thinks the potential for model improvement outweighs the concern.

Steele begins building a model that combines the structured financial data and the sentiment data. She starts with cleansing and wrangling the raw structured financial data. Exhibit 1 presents a small sample of the raw dataset before cleansing: Each row represents data for a particular firm.

Exhibit 1 Sample of Raw Structured Data Before Cleansing

ID	Ticker	IPO Date	Industry (NAICS)	EBIT	Interest Expense	Total Debt
1	ABC	4/6/17	44	9.4	0.6	10.1
2	BCD	November 15, 2004	52	5.5	0.4	6.2
3	HIJ	26-Jun-74	54	8.9	1.2	15.8
4	KLM	14-Mar-15	72	5.7	1.5	0.0

After cleansing the data, Steele then preprocesses the dataset. She creates two new variables: an "Age" variable based on the firm's IPO date and an "Interest Coverage Ratio" variable equal to EBIT divided by interest expense. She also deletes the "IPO Date" variable from the dataset. After applying these transformations, Steele scales the financial data using normalization. She notes that over the full sample dataset, the “Interest Expense” variable ranges from a minimum of 0.2 and a maximum of 12.2, with a mean of 1.1 and a standard deviation of 0.4.

Steele and Schultz then discuss how to preprocess the raw text data. Steele tells Schultz that the process can be completed in the following three steps:

Step 1 Cleanse the raw text data.

Step 2 Split the cleansed data into a collection of words for them to be normalized.

Step 3 Normalize the collection of words from Step 2 and create a distinct set of tokens from the normalized words.

With respect to Step 1, Steele tells Schultz:

> “I believe I should remove all html tags, punctuations, numbers, and extra white spaces from the data before normalizing them.”

After properly cleansing the raw text data, Steele completes Steps 2 and 3. She then performs exploratory data analysis. To assist in feature selection, she wants to create a visualization that shows the most informative words in the dataset based on their term frequency (TF) values. After creating and analyzing the visualization, Steele is concerned that some tokens are likely to be noise features for ML model training; therefore, she wants to remove them.

Steele and Schultz discuss the importance of feature selection and feature engineering in ML model training. Steele tells Schultz:

> “Appropriate feature selection is a key factor in minimizing model overfitting, whereas feature engineering tends to prevent model underfitting.”

Once satisfied with the final set of features, Steele selects and runs a model on the training set that classifies the text as having positive sentiment (Class “1” or negative sentiment (Class “0”). She then evaluates its performance using error analysis. The resulting confusion matrix is presented in Exhibit 2.

Exhibit 2 Confusion Matrix

		Actual Training Results	
		Class “1”	Class “0”
Predicted Results	Class “1”	TP = 182	FP = 52
	Class “0”	FN = 31	TN = 96

1 Which of Steele’s statements relating to the steps in building structured data-based and text-based ML models is correct?

A Only Statement 1 is correct.

B Only Statement 2 is correct.

C Statement 1 and Statement 2 are correct.

2 Steele’s concern about using Twitter data in the model *best* relates to:

A volume.

B velocity.

C veracity.

3 What type of error appears to be present in the IPO Date column of Exhibit 1?

A invalidity error.

B inconsistency error.

C non-uniformity error.

4 What type of error is most likely present in the last row of data (ID #4) in Exhibit 1?

A Inconsistency error

B Incompleteness error

C Non-uniformity error

5 During the preprocessing of the data in Exhibit 1, what type of data transformation did Steele perform during the data preprocessing step?

A Extraction

B Conversion

C Aggregation

6 Based on Exhibit 1, for the firm with ID #3, Steele should compute the scaled value for the "Interest Expense" variable as:

A 0.008.

B 0.083.

C 0.250.

7 Is Steele's statement regarding Step 1 of the preprocessing of raw text data correct?

A Yes.

B No, because her suggested treatment of punctuation is incorrect.

C No, because her suggested treatment of extra white spaces is incorrect.

8 Steele's Step 2 can be *best* described as:

A tokenization.

B lemmatization.

C standardization.

9 The output created in Steele's Step 3 can be *best* described as a:

A bag-of-words.

B set of n-grams.

C document term matrix.

10 Given her objective, the visualization that Steele should create in the exploratory data analysis step is a:

A scatter plot.

B word cloud.

C document term matrix.

11 To address her concern in her exploratory data analysis, Steele should focus on those tokens that have:

A low chi-square statistics.

B low mutual information (ML) values.

C very low and very high term frequency (TF) values.

12 Is Steele's statement regarding the relationship between feature selection/feature engineering and model fit correct?

A Yes.

B No, because she is incorrect with respect to feature selection.

C No, because she is incorrect with respect to feature engineering.

13 Based on Exhibit 2, the model's precision metric is *closest* to:

A 78%.

B 81%.

C 85%.

14 Based on Exhibit 2, the model's F1 score is *closest* to:

A 77%.

B 81%.

C 85%.

15 Based on Exhibit 2, the model's accuracy metric is *closest* to:

A 77%.

B 81%.

C 85%.

SOLUTIONS

1 B is correct. The five steps in building structured data-based ML models are: 1) conceptualization of the modeling task, 2) data collection, 3) data preparation and wrangling, 4) data exploration, and 5) model training. The five steps in building text-based ML models are: 1) text problem formulation, 2) data (text) curation, 3) text preparation and wrangling, 4) text exploration, and 5) model training. Statement 1 is incorrect: Text preparation and wrangling is the third step in building text ML models and occurs after the second data (text) curation step. Statement 2 is correct: The fourth step in building both types of models encompasses data/text exploration.

2 C is correct. Veracity relates to the credibility and reliability of different data sources. Steele is concerned about the credibility and reliability of Twitter content, noting that research suggests that as much as 10%–15% of social media content is from fake accounts.

3 C is correct. A non-uniformity error occurs when the data are not presented in an identical format. The data in the "IPO Date" column represent the IPO date of each firm. While all rows are populated with valid dates in the IPO Date column, the dates are presented in different formats (e.g., mm/dd/yyyy, dd/mm/yyyy).

4 A is correct. There appears to be an inconsistency error in the last row (ID #4). An inconsistency error occurs when a data point conflicts with corresponding data points or reality. In the last row, the interest expense data item has a value of 1.5, and the total debt item has a value of 0.0. This appears to be an error: Firms that have interest expense are likely to have debt in their capital structure, so either the interest expense is incorrect or the total debt value is incorrect. Steele should investigate this issue by using alternative data sources to confirm the correct values for these variables.

5 A is correct. During the data preprocessing step, Steele created a new "Age" variable based on the firm's IPO date and then deleted the "IPO Date" variable from the dataset. She also created a new "Interest Coverage Ratio" variable equal to EBIT divided by interest expense. Extraction refers to a data transformation where a new variable is extracted from a current variable for ease of analyzing and using for training an ML model, such as creating an age variable from a date variable or a ratio variable. Steele also performed a selection transformation by deleting the IPO Date variable, which refers to deleting the data columns that are not needed for the project.

6 B is correct. Steele uses normalization to scale the financial data. Normalization is the process of rescaling numeric variables in the range of [0, 1]. To normalize variable X, the minimum value (X_{min}) is subtracted from each observation (X_i), and then this value is divided by the difference between the maximum and minimum values of X ($X_{\text{max}} - X_{\text{min}}$):

$$X_{i\ (\text{normalized})} = \frac{X_i - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

The firm with ID #3 has an interest expense of 1.2. So, its normalized value is calculated as:

$$X_{i\ (\text{normalized})} = \frac{1.2 - 0.2}{12.2 - 0.2} = 0.083$$

7 B is correct. Although most punctuations are not necessary for text analysis and should be removed, some punctuations (e.g., percentage signs, currency symbols, and question marks) may be useful for ML model training. Such punctuations should be substituted with annotations (e.g., /percentSign/, /dollarSign/, and /questionMark/) to preserve their grammatical meaning in the text. Such annotations preserve the semantic meaning of important characters in the text for further text processing and analysis stages.

8 A is correct. Tokenization is the process of splitting a given text into separate tokens. This step takes place after cleansing the raw text data (removing html tags, numbers, extra white spaces, etc.). The tokens are then normalized to create the bag-of-words (BOW).

9 A is correct. After the cleansed text is normalized, a bag-of-words is created. A bag-of-words (BOW) is a collection of a distinct set of tokens from all the texts in a sample dataset.

10 B is correct. Steele wants to create a visualization for Schultz that shows the most informative words in the dataset based on their term frequency (TF, the ratio of the number of times a given token occurs in the dataset to the total number of tokens in the dataset) values. A word cloud is a common visualization when working with text data as it can be made to visualize the most informative words and their TF values. The most commonly occurring words in the dataset can be shown by varying font size, and color is used to add more dimensions, such as frequency and length of words.

11 C is correct. Frequency measures can be used for vocabulary pruning to remove noise features by filtering the tokens with very high and low TF values across all the texts. Noise features are both the most frequent and most sparse (or rare) tokens in the dataset. On one end, noise features can be stop words that are typically present frequently in all the texts across the dataset. On the other end, noise features can be sparse terms that are present in only a few text files. Text classification involves dividing text documents into assigned classes. The frequent tokens strain the ML model to choose a decision boundary among the texts as the terms are present across all the texts (an example of underfitting). The rare tokens mislead the ML model into classifying texts containing the rare terms into a specific class (an example of overfitting). Thus, identifying and removing noise features are critical steps for text classification applications.

12 A is correct. A dataset with a small number of features may not carry all the characteristics that explain relationships between the target variable and the features. Conversely, a large number of features can complicate the model and potentially distort patterns in the data due to low degrees of freedom, causing overfitting. Therefore, appropriate feature selection is a key factor in minimizing such model overfitting. Feature engineering tends to prevent underfitting in the training of the model. New features, when engineered properly, can elevate the underlying data points that better explain the interactions of features. Thus, feature engineering can be critical to overcome underfitting.

13 A is correct. Precision, the ratio of correctly predicted positive classes (true positives) to all predicted positive classes, is calculated as:

$$\text{Precision (P)} = \text{TP}/(\text{TP} + \text{FP}) = 182/(182 + 52) = 0.7778\ (78\%).$$

14 B is correct. The model's F1 score, which is the harmonic mean of precision and recall, is calculated as:

$$\text{F1 score} = (2 \times \text{P} \times \text{R})/(\text{P} + \text{R}).$$

$$\text{F1 score} = (2 \times 0.7778 \times 0.8545)/(0.7778 + 0.8545) = 0.8143\ (81\%).$$

15 A is correct. The model's accuracy, which is the percentage of correctly predicted classes out of total predictions, is calculated as:

Accuracy = (TP + TN)/(TP + FP + TN + FN).

Accuracy = (182 + 96)/(182 + 52 + 96 + 31) = 0.7701 (77%).