Table of Contents

Introduction ........................................................................................................................................... 3
Privacy as a Driver of Data Security ....................................................................................................... 4
Overfitting: A Security Perspective ....................................................................................................... 6
Data as a Protected Asset ....................................................................................................................... 7
Is Security a Priority on Analytics Projects? ......................................................................................... 7
Computing a False Positive ................................................................................................................... 8
Common Attacks in Machine Learning ................................................................................................. 9
  Random Seed Attacks .......................................................................................................................... 9
  Poisoning Attacks ............................................................................................................................. 9
  Missing Values and Bad Data Attacks ................................................................................................ 11
  White Noise Attacks ........................................................................................................................ 13
  No-SQL Injection ............................................................................................................................ 13
Protecting Your Analytics Environment ............................................................................................... 13
Understanding Attack Vectors ............................................................................................................. 14
Conclusion ............................................................................................................................................... 15
Bibliography ......................................................................................................................................... 15

Disclaimer: The views, processes or methodologies published in this article are those of the author. They do not necessarily reflect Dell EMC’s views, processes or methodologies.
Introduction

Data scientists rarely consider the risks associated with using certain data sets that possibly could contain “poison inputs”, or in other words, specially crafted outliers designed to control the outcome of an analytics system. Practitioners of data science are often at the front lines of decision making where the results of our analytics experiments have a “vote” in the output of organizational decisions. If there is a real-world consequence of this decision, then it should be our responsibility to protect the integrity of the analytics software and decision support systems by first considering the potential attack vectors and then implementing measures (whether that be in code or in process) to address these weaknesses and mitigate the risk of the models being technically correct but exploited, yielding invalid results.

Classifiers are ubiquitous in machine learning and are used to assign one of two labels (called a binary classifier) to a set of feature vectors. For example, an image hosting website might use a classifier to decide whether an image (represented as a matrix of pixels) submitted to their service is “prohibited” or “allowed”. A creative attacker can force the software into recognizing images that should be prohibited images as allowed [1]. In the photo below, the top image has been altered slightly by inserting random pixels in the top left-hand corner. This is enough to change the classification of the image from ‘allowed’ to ‘prohibited’.

One class of “intelligent” software particularly vulnerable to these attacks are decision support systems. A decision support system is a collection of analytics programs and other programs forming a system that can accept some input data and produce an output that aids in decision making. Decision support systems may include classifiers and other machine learning software as the analytics programs. The goal of this article is to raise the reader’s awareness on the attack vectors that are specific to these analytics programs.
which may run in a combination of NoSQL datastores, relational database management systems (RDBMS) or as standalone scripts that implement machine learning functionality. These scripts are often written in domain-specific languages, such as Spark.

The systems that are implemented in RDBMs using in-database analytics are often more secure than applications written in NoSQL datastores which may not have native support for authentication or authorization. These applications are especially vulnerable to NoSQL injection and man-in-the-middle attacks [2].

Decision support systems are particularly vulnerable to attacks because they are often on the front line between decisions being made and data. The ability motivation for a malicious user or group to control the knowledge value-chain process within an organization may financial or political gain.

A sophisticated decision support system can be manipulated by the environment in which it operates. In fact, adaptability and flexibility are two requirements for decision support systems and there is a trade-off between security and adaptability that needs to be considered in the initial design stages [2].

The consequences of an attack on a decision support system can be costly decisions that are unsupported by facts and can be just as costly as down-time in your system. Aberdeen Research conducted a study that found downtime costs an organization an average of $138,000 per hour [3]. However, the amount most organizations invest in high availability and disaster recovery solutions eclipses the amount budgeted for protecting decision support systems from being exploited by poison inputs, malicious attacks or internal misuse and abuse by information workers who understand how analytics applications work internally.

Throughout this article we use the term “exploited system” to describe a decision-making system that has been exploited in such a way that an attacker has some control over the outputs of the system and discuss ways this escalated control over the decision process which was not intended by the designers of the system can be achieved. It is the goal of this article to walk the reader through some common attack vectors of decision making systems, and to encourage greater levels of consideration from experts in the fields of both data science and information security.

**Privacy as a Driver of Data Security: A Case Study**

A recent project involving the design and implementation of a sentiment analysis system perfectly demonstrates the thesis that privacy is a driver of security. The goal of the project was to utilize the Twitter public streaming API and determine the overall sentiment of a retail brand from the stream. This is a common problem in marketing research and is often used to support business applications such as “360-degree views of customer” dashboards.

As part of the case study, which involved building and analyzing sensitive customer data, we rated 3 popular options for building a decision support system that employed SVMs (Support Vector Machines) on a large training corpus we had pulled from the public API. Table 1 lists some features of each option considered, including privacy. Ultimately, since the business intended on mashing the Twitter data with internal customer data, they opted for the more expensive solution which provided the greatest promise of privacy for the customer.
Privacy is a very important requirement and can be weighted equally as heavily as scalability, speed or availability requirements. A decision support system that provides privacy must have some level of security built-in and so we see how privacy can be used to drive the conversation towards data security and the hardening of the decision system.

<table>
<thead>
<tr>
<th>Privacy Options</th>
<th>PySpark</th>
<th>TextBlob</th>
<th>IBM Watson</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>Fast</td>
<td>Slow</td>
<td>Fast</td>
</tr>
<tr>
<td>Accuracy</td>
<td>Accurate</td>
<td>Accurate</td>
<td>Accurate</td>
</tr>
<tr>
<td>Real-Time</td>
<td>Yes, if used with Spark Streaming instance</td>
<td>No</td>
<td>Yes, if used with Spark Streaming API</td>
</tr>
<tr>
<td>Scalability</td>
<td>High</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Pricing</td>
<td>Free</td>
<td>Free</td>
<td>Cloud pricing</td>
</tr>
<tr>
<td>Customizability</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>

Table 1: Common requirements for decision support systems

In addition to privacy, the client was concerned with use of public data streams in the analysis. The use of Tweets which could be created by anyone (even a computer) poses the risk that bad quality data may find its way into the analysis. With no way to tell if a Tweet is from a customer or someone trying to sabotage a brand on Twitter, it was vital to monitor the accuracy of our algorithm not just once but in real-time. Our solution was to develop a custom dashboard pictured in Figure 2 that displayed a confusion matrix for our classifier. This enabled the users to monitor the integrity of the system in real-time by refreshing the dashboard.
Remembering that businesses value customer privacy is an important point to mention when obtaining executive buy-in for your data security project. It is much easier to convince most managers that customer privacy is worth protecting than it would be to convince them of the merits of investing in security of the system.

**Overfitting: A Security Perspective**

I hinted at one of the most common examples in the field of machine learning security: deep convolutional networks. Computer vision systems that classify images can be easily tricked into either accepting or rejecting an image as “prohibited” or “allowed”, for instance by changing an area of the image (say a few hundred pixels from green to red), forcing the image over the decision boundary defined by the classifier and flipping the output from “prohibited” to “allowed” or vice versa.

The reason these systems are vulnerable to this type of attack is because the algorithms tend to segment the image into multiple pieces and analyze each one individually, a kind of divide-and-conquer strategy that is common in data science. The consequence is that any slight change in a group of pixels – i.e. the upper left corner of the image – can cause the entire image, which may be close to the decision boundary, to be pushed over the decision boundary, essentially flipping the output bit.

Consider the ability to apply this to a bank’s credit scoring software. Using this method, an individual with credit just under the acceptance rate for a loan could potentially change the decision by artificially increasing one of the other attributes being processed by the deep artificial neural network’s (ANN) hidden layer.
This hidden layer is the internal memory of the ANN model and represents the model’s “understanding,” or more technically the internal representation, of the data. However, this representation of the features of the data, encoded as connections between various nodes in a network which may be strong or weak (known as “weights” that may be nearly any real number), is not very amicable to interpretation.

This poses some challenge in testing a software that uses an ANN. Blackbox testing is the only option since it is likely not possible to find an interpretation of every weight in the hidden layer, especially if the model consists of hundreds of nodes.

One risk associated with this is overfitting (or underfitting), but this risk is often not discussed in a security context. However, having arbitrary features in the relationship between nodes could be used by an attacker to flip the output, as demonstrated in the computer vision example. If an attacker knows the model has given weight to an attribute that he or she can control, then the system is at risk. This control may not have been intentionally added by the designers of the model, but derived from the limitations of blackbox testing.

Many analytics projects in industries where security is not as high of a concern as in the banking industry may not thoroughly test their application, leaving it vulnerable to these types of simple exploits that are inherent to most learning algorithms.

One way to ensure this type of attack is difficult to pull off would be to choose a different type of model, for example a decision tree where the rules employed by the model have unambiguous interpretations that can be validated by business rules. This most likely will come at the expense of accuracy in the model, and so we can see that hardening the decision support system does not always come for free. Another option would be to choose an ensemble of models and take the average (also known as “bagging”). These data science techniques help to decrease the chance of overfitting, which not only is good for the accuracy of the models but can also eliminate a key weakness from a security standpoint.

In general, how sensitive the model is to changes in the data is an important security consideration and should be evaluated against the requirements of the project. Slight changes in the training data can cause the decision boundary of the model to change drastically, potentially creating a security vulnerability that could be exploited in the system. The model should be tested thoroughly for this kind of sensitivity.

Data as a Protected Asset

Raw data can have a monetary value especially if it cannot be replaced and is essential to the success of the project. Hence, it makes sense that data should be safeguarded as valuable assets. This means using encryption to protect them from exposing sensitive information for cases where the analytics questions being asked are highly sensitive. Checksums can ensure the integrity of data that may be vulnerable to corrupted file systems, software errors, errors in transporting the data, or a malicious user modifying the data.

Is Security a Priority on Analytics Projects?

It's been estimated that 24 hours of downtime costs financial companies 1.3 million dollars per day. [3] This can be caused by cyber attacks, from denial of services to malware designed to cause damage to critical data infrastructure or steal customer information. Traditional data-driven applications seek to
mitigate these issues of data privacy breaches and hardening data, but for many data scientists, security never makes it into the requirements or the budget. [4]

Often data scientists come from an analytics background rather than a security background, and as such they may not be aware of weaknesses in their decision support system and tend to focus more on the statistical validity of the model. However, if the integrity of a model has been compromised, the output is invalid regardless of the technical correctness of the analysis of the statistics used.

One of the most important but often overlooked attributes of a successful analytics project is that it is trusted by the business in question. Too often analytics projects are technically correct but fail to be trusted by the business. One way to safeguard the integrity of your system is to monitor the error rate of each module where it makes sense to do so. For example, if your decision support system uses classifiers it makes sense to compute this and then monitor it. If you already do this, you might consider implementing some statistical control processes around your system so you can ensure if the error rate goes beyond some chosen threshold, users are alerted (or if it’s a control system, it may be better to automatically turn it off). Having statistical control of your error rate preserves the integrity of the system [4].

Regular health checks on the error rate can make it more difficult for attackers to compromise your system. A successful attack would have to keep the error rate in control, although this is possible to do by computing false positives (or false negatives) as we will describe in the next section.

**Computing a False Positive**

Many data scientists would prefer to spend their time reducing the error rate of their models rather than testing them for security vulnerabilities or ways they could be abused. A simple way to check the health of the model is to look at its error rate and monitor this value. However, a piece of malware that affects an analytics system may be programmed to report an error rate that makes the analytics team believe the software is generating meaningful results, leaving the malware undetected. It is thus important to make sure that the module which calculates the error rate for the project is protected.

Depending on the level of confidence required to ensure that the analytics software is not exploited, further lengths must be taken to maintain the integrity of the error rate. For example, data scientists should understand how the true positive/false positive rate of a classification system can be affected by various inputs (we will use this in the next section to describe an original attack using missing data).

If the data collection process is controlled, we may have more confidence in the output. However, if the false negative rate should be kept to a minimum, for example in medical applications where the specificity of the test is critical, this metric should also be monitored. If not, a malicious user could simply increase the false negative rate while minimizing the false positive by a proportional amount, cleverly ensuring that the overall error rate remains the same and the attack is unnoticed.

In the context of a medical application this may be obvious, but for other applications it may be that manipulating the false positive and false negative rate one way or the other gives the attacker some leverage in creating sophisticated attacks.

If the system has a higher than usual false positive rate, the attacker may choose their inputs to create bias toward the attacker’s goal decision. More information on how false positives can be computed can
be found here [5]. We will now look at some common attacks in machine learning and end by describing an original attack using missing data.

**Common Attacks in Machine Learning**

**Random Seed Attacks**

Random seeds are extremely important when testing analytics systems, especially those that are non-deterministic and rely on randomly generated input (such as Monte Carlo simulation). Random seeds are used by data science teams to reproduce results during testing and design phases of the project and may be left in shared data science notebooks like Jupiter which may be published publicly on version control systems like Github or Bitbucket. If these seeds become known to an attacker or are used in production code by accident, they can be used to compute exactly which inputs will result in false positives (or true positives).

To execute a maneuver like this, it would be necessary for the attacker to know exactly which set of inputs or hyperparameters of the model map to which decision and then carefully crafting inputs (assuming the data stream being analyzed can be controlled by the attacker). Some classifiers that use deep convolutional networks are prone to this type of attack because changing an input slightly tends to result in a sharp change in the decision output. [8] This means computing a false positive is as simple as taking an image that should score negative and changing it slightly (maybe running it through another convolutional network) to flip the decision output. Using models that are based on continuous functions such as logistic regression may be less prone to this attack.

Hackers will always seek to push the boundaries of what can and cannot be done. At the time of writing (January 2018), attacks of this nature are not very common but to think that this idea is new or original would be a mistake and probably naïve. There will always be a motivation to manipulate the outcome of decision support systems in favor of certain decisions whether it be financial, political, or both.

It is well-known that using public data carries inherent risks. However, using publicly available data, or data that can potentially be manipulated in a data science workflow, is often a necessary risk as this data contains the most information about the real world; it may also contain information about the behavior and patterns of your customers that would not be observed if a controlled experiment or simulation were used.

**Poisoning Attacks**

The idea behind a poisoning attack is simple. If a program accepts user input, even if it is “sanitized” and the code makes use of a training dataset that depends on this user input, it is possible for a malicious user to inject specially-crafted inputs designed to increase the error rate of the system. The malicious user injects outliers in the system that may be specially-crafted to avoid detection as “bad inputs”. [6]

Take clustering, for example. This is traditionally an unsupervised machine learning problem, meaning that it does not require training data and can even be modified to be “online” meaning it will work on a stream of data. Let’s pretend our data stream is a streaming API that takes “real” data from a video stream and is the basis for many monitoring software systems that can detect changes between frames by performing clustering. This may be the basis for AI built into your closed-circuit video monitoring solution.
As for the inherent weakness, it should be noted that it is not even necessary to disable the camera if they wanted to trespass; all they would have to do is disable the “AI” layer. This is easily accomplished by injecting outliers into the video stream. For example, setting up lasers in fixed positions that were encoded as centroids by the AI software would allow the attacker to control the output of the algorithm.

![Elbow Curve](image)

**Figure 2: Elbow Method in Clustering Applications**

An elbow curve such as the one generated in R above are typically used in the early stages of the analytics workflow as a “model selection” criterion where the model with the lowest value of k (number of clusters) is chosen. The theory is that the higher the explained variance, the better the model. However, a property of this class of clustering models is that the explained variance can be increased by simply increasing the value of k. To offset this, data scientists will look for the “elbow” in the curve (around k = 3 clusters in the one above).

Clustering has been used in malware detection algorithms but recently the results have been criticized by security experts as clustering algorithms are prone to a poisoning attack [7]. A malicious attacker may understand the model selection criterion and be able to manipulate it so the system manager may choose the wrong model, or choose one of the attacker’s choice. In the case of intrusion detection systems, using clustering algorithms are a poor idea because of how sensitive clustering algorithms are to poison inputs.

How is this done in practice? In the case of a bottom-up algorithm like k-means, the elbow curve method is often used as a decision rule. An elbow curve such as the one pictured above can easily be manipulated by injecting outliers, causing the data to appear to have many more clusters. Even one additional cluster may influence the ultimate decision being made. Attacks against clustering algorithms often involve
poisoning inputs and clustering algorithms remain vulnerable even when only a small percentage of the input can be controlled by an attacker. [7]

Missing Values and Bad Data Attacks

Missing data or invalid data can skew results. Missing values may be a consequence of “bad data” when users of the system are permitted to enter free-form data upstream. Another possibility that is rarely considered is missing values can be inserted into the data by an attacker with a simple SQL query. Depending on how the system deals with these missing values will determine if the attack is able to successfully affect the integrity of the system either by decreasing the information available or changing fundamental parameters of the data such as variance and mean.

There should be some code in place that handles missing values and invalid data. The risk of removing it is the loss of valuable information. The risk of using it as-is is skewed results. This differs from the traditional problem of sanitizing inputs, as missing values may be acceptable or invalid characters if they don’t contain certain prohibited characters. On the other hand, analytics models are far more sensitive to user input, requiring not only properly structured data but the intrinsic parameters of the data distribution – such as mean and variance – must not change much or it will affect the analysis – garbage in, garbage out. For this reason, careful consideration must be given to how to deal with data quality issues and how to handle them in the analysis.

Attacks that use missing data are particularly sophisticated because they can modify the underlying distribution of the data by skewing the mean to the left or right (green distribution in Figure 3 below). Readers familiar with statistics might note that missing values can be replaced with the mean of all “similar” values which would preserve the mean of the distribution regardless of the number of missing values. However, this procedure changes the variance. Sophisticated algorithms such as imputation are used to remove the missing values while preserving both the variance and the mean of the data distribution.

The two narrow-long distributions in Figure 3 represent normally distributed data with the same mean but slightly different variances. Imputation can intelligently keep the variance the same as the original data to avoid this attack.
The question that should be asked is: Are the missing values in the data set missing because of some known reason, i.e. a malicious user, or do they occur at random? For instance, is removing 5 - 10% of the values from a large sample reasonable without decreasing the information available to solve your problem?

The decision to remove missing values has consequences. The most obvious of which is our analysis can become less accurate if by dropping missing data we remove information that would have been useful in our analysis. Care must be taken when removing missing values but statistical software package such as R’s MICE can be used to perform the technique of imputation on both categorical and numerical data which can preserve original parameters of the data distribution regardless of missing values [8].
White Noise Attacks

The idea of a “white noise attack” is simple. An attacker simply injects bad data. This is like the attack we described using missing values except the intent is not to just skew the distribution and mislead analysis. Rather, the intent is to decrease the information content of the data.

Since analytics workflows tend to store raw data that may be poor quality, there may be no obvious way to decide which data to include or exclude in the training data. When making use of data streams or polling public data sources as inputs to an analytics system, it would be a mistake to include all training data. Unfortunately, as data scientists we often want to include as much data as possible; it can be challenging to resist the temptation to use all data when data is scarce in the first. However, if we have no control or understanding of the statistical distribution that underlies the data we’re studying, we risk accepting noise into our workflow which could potentially be used to control the decision support system.

The Extract, Transform, Load (ELT) process should have some mechanism to remove white noise from the data. Fortunately, detecting outliers, including missing data, is well understood and can easily be implemented. One way to do this in many statistical languages (R and SAS for instance) would be to remove the upper and lower tails of the data. Missing data can also skew results and is easy for an attacker to inject into the data where it may not be detected as “white noise” data at all. Fortunately, it is straightforward to implement this in SAS and R.

No-SQL Injection

SQL injection is a familiar term for any kind of web development. Many business intelligence systems are vulnerable to SQL injection attacks, including the underlying reporting software which may generate SQL. It is important then to consider the reporting interface as a potential attack vector. Vendors of analytics reporting software may not place as much importance on security as data scientists, and it should be an important consideration when choosing the reporting tool used by users when accessing the analytic models.

In most cases, data warehouses are built using relational database technology such as Microsoft SQL Server or Oracle. Database administrators are aware of security risks and have likely locked down the source data with proper authorization, authentication and role based security; however, in the case of analytics models which may not be deployed to a secure relational database, it is important to harden the data stores where models are stored so they cannot be changed by someone who isn’t authorized to do so. A security matrix should be a necessary document in every analytics project where there may be reason to want to manipulate the decision.

In cases where a NoSQL database (MongoDB, CouchDB, etc.) is used as the data layer of the decision support system, you should be especially concerned with cross-site scripting attacks and NoSQL injection. This is because NoSQL databases are still an emerging technology and security is in its infancy. It is recommended that authorization and authentication modules are written to protect data assets from the poisoning attacks discussed earlier, even if it means writing custom code. For a comprehensive analysis of NoSQL injection attacks, see [9].

Protecting Your Analytics Environment

The idea that security is the sole responsibility of a small group of IT professionals that manage security assets for the organization is incorrect. Security of the analytics project should be the responsibility of
the analytics team, since IT cannot be expected to understand the complex relationship between the finely-tuned hyperparameters of analytics systems and the error function. Understanding under what conditions of those parameters could maliciously lower the error rate of the system is an important consideration. Tight control over the underlying distribution of the training data and some process that handles rejection of observations that are outliers but may appear close enough to the mean may remain undetected. This requires application of statistical control to the underlying raw data streams being analyzed.

So how do we secure our analytics projects when public data streams are used? Again, the importance of understanding the underlying distribution of the training data set is the key to controlling the error rate of the system. What does this mean in practical terms? Having some practical idea of what data is expected, and what is not expected, can help mitigate this risk. Using a dashboard to monitor key metrics like false-positive and false-negative rate is helpful.

Other than monitoring the health of your environment regularly, authentication and authorization should always be used especially if your analytics sandbox is built on top of a NoSQL datastore. It should be a priority to check that the datastore you’re using supports authentication and if not, it may require writing custom code. Remember that poisoning attacks are a common theme when compromising datastores so protecting the integrity of the data is vital to keep trust of the business in the decision support / analytics system.

A fool-proof way to ensure data integrity is by using simulations rather than real data whenever possible. If random data is simulated each time, the chance of a poisoning attack decreases. Remember to remove random seeds from any published code to keep the simulation truly random. If an attacker can repeat the simulation then randomness no longer provides any protection.

**Understanding Attack Vectors**

One way to counteract attacks against analytics software is to monitor the error rate of analytics projects to ensure their ongoing health. It must be noted, however, that simply monitoring this error rate does not mean the analytics system has not been exploited for malicious intent. Pre-emptive action can be taken to ensure that the software is free of common bugs, written in a secure way, and that the most common type of exploits against software have been considered in the design of the system.

The following is a checklist of what not to do when it comes to designing the most secure analytics software. As data scientists, we should be asking ourselves if any of these apply to the systems we create, evaluate the risks and, if necessary, take steps to mitigate these risks.

- Not rejecting outliers in data
- Accepting free form user input
- Not monitoring the error rate periodically
- Not measuring the error rate at all
- Using the “wrong” type of error calculation (example, only using accuracy in a classification system, not paying attention to false positive rate)
- Protecting against white noise attacks that reduce error rate
- Using public data streams that could be controlled by attackers
- Not securing the underlying source systems
• Not securing the ELT (extract, load transform) software
• Lack of RBAC in the analytics sandbox
• Failure to consider SQL injection attacks in code
• Failure to consider NO-SQL Injection attacks in code
• Outlier Detection if using public data streams (social media analysis)

Conclusion
The long-term success or failure of an analytics project depends critically on the trust the users place on the system to make sound decisions. If the system is to be truly trusted, it should be not only free from defects that could be exploited, but it should also be monitored to ensure the error rate of the system and other critical measures of the health of the system don’t suddenly fall outside expected bounds. If the system is not healthy by this definition, then it may be a symptom that the analytics system has been exploited.

More generally, the importance of statistical process control, where necessary to mitigate the risk of attacks on publicly streamed training data or training data that could potentially be modified by an adversary, cannot be overstated.

As decision making systems that utilize machine learning and AI software continue to proliferate our lives, their potential to be exploited for financial or political gain increases. It is thus critical that we understand the specific weaknesses and attack vectors of analytics systems to mitigate the risk of an adversary being able to not only invalidate the results but possibly even control the decision-making process completely.

It is critical for practitioners of data science to step outside their analytics background and consider the security implications if they are to design secure analytics software that can be trusted by the decision makers and uses of the system.

Bibliography


