Proyecto de **Ciberseguridad** con **Machine Learning**

→Protección contra ataques de fuerza bruta.

INDICE

Muestreo aleatorio ** Eliminar atributos altamente correlacionados ** Aplicar reducción de dimensionalidad

2.3 Transformación de Datos

2.3.1 Crear transformaciones lineales y no lineales de todos los atributos * Cuadrado * Raíz cuadrada * Estandarizar * Normalizar * Discretizar

2.3.2 Descomponer atributos complejos en sus partes constituyentes

3. Evaluar Algoritmos

3.1 Crear un Conjunto de Pruebas

Crear un conjunto de validación para su posterior uso Evaluar y seleccionar una opción de prueba apropiada * Conjuntos de entrenamiento y prueba * Validación cruzada k-fold Seleccionar una medida de rendimiento utilizada para evaluar los modelos

3.2 Evaluar Algoritmos Candidatos

Seleccionar un conjunto diverso de algoritmos para evaluar (10-20) Utilizar configuraciones de parámetros de algoritmos comunes o estándar

4. Mejorar los Resultados

4.1 Ajuste del Algoritmo

Utilizar parámetros de modelo históricamente efectivos Explorar el espacio de parámetros del modelo Optimizar los parámetros del modelo con buen rendimiento

4.2 Métodos de Conjunto

Utilizar bagging en modelos con buen rendimiento Utilizar Boosting en modelos con buen rendimiento Mezclar los resultados de modelos con buen rendimiento

4.3 Selección del Modelo

Seleccionar un subconjunto diverso de modelos con buen rendimiento (5-10) Evaluar modelos con buen rendimiento en un conjunto de validación Seleccionar un pequeño grupo de modelos con buen rendimiento (1-3)

1. **Definir el Problema:**

1.1 ¿Cuál es el problema?

1.1.1 Definir el problema informalmente: El problema consiste en la vulnerabilidad de sistemas informáticos a ataques de fuerza bruta, donde un atacante intenta adivinar credenciales válidas probando diferentes combinaciones de usuarios y contraseñas hasta encontrar la correcta.

1.1.2 Definir el problema formalmente: El problema consiste en desarrollar un sistema de detección y prevención de ataques de fuerza bruta que pueda identificar patrones de tráfico malicioso en la red y tomar medidas para bloquear o mitigar el impacto de estos ataques.

1.1.3 Enumerar las suposiciones sobre el problema:

- Es considerado que los ataques de fuerza bruta representan una amenaza significativa para la seguridad de los sistemas informáticos.
- Se presupone que el sistema dispondrá de datos históricos de tráfico para entrenar el modelo de machine learning.
- Se toma en cuenta la necesidad de que el sistema pueda adaptarse a nuevos métodos y estrategias utilizadas por los atacantes.

1.1.4 Enumerar problemas similares:

- Detección de intrusiones en sistemas informáticos.
- Prevención de ataques de denegación de servicio (DDoS).
- Protección contra ataques de phishing y malware.

1.2 ¿Por qué necesita ser resuelto el problema?

1.2.1 Describir la motivación para resolver el problema: Es crucial resolver este problema porque los ataques de fuerza bruta representan una amenaza seria para la seguridad de la información. Pueden comprometer la integridad, confidencialidad y disponibilidad de los datos, lo que puede tener consecuencias graves para individuos y organizaciones.

1.2.2 Describir los beneficios de la solución (predicciones del modelo): La solución permitirá detectar y mitigar los ataques de fuerza bruta de manera más rápida y efectiva, protegiendo así los sistemas y datos sensibles. Además, reducirá el riesgo de intrusiones no autorizadas y el tiempo y costos asociados con la recuperación después de un ataque.

1.2.3 Describir cómo se utilizará la solución: La solución se integrará en los sistemas de seguridad existentes de una organización para mejorar la capacidad de detección y respuesta a los ataques de fuerza bruta. Se ejecutará en tiempo real para monitorear el tráfico de red y tomar medidas proactivas para bloquear o mitigar los ataques.

1.3 ¿Cómo podría resolverse el problema manualmente?

1.3.1 Describir cómo se resuelve actualmente el problema (si es que se resuelve): Actualmente, el problema se aborda mediante la implementación de medidas de seguridad como el bloqueo de direcciones IP después de un cierto número de intentos fallidos de inicio de sesión. También se pueden utilizar listas de bloqueo para evitar que las direcciones IP conocidas por realizar ataques de fuerza bruta accedan al sistema.

1.3.2 Describir cómo un experto en la materia haría predicciones manuales: Un experto en ciberseguridad puede analizar los registros de tráfico de red en busca de patrones sospechosos que puedan indicar un ataque de fuerza bruta. Este análisis manual requiere experiencia en la identificación de comportamientos anómalos y la capacidad de tomar decisiones rápidas para mitigar la amenaza.

1.3.3 Describir cómo un programador podría codificar manualmente una solución: Un programador podría desarrollar scripts o herramientas personalizadas para monitorear y analizar el tráfico de red en busca de patrones asociados con ataques de fuerza bruta. Estos scripts podrían incluir reglas específicas para identificar intentos de inicio de sesión fallidos repetidos y tomar medidas para bloquear o limitar el acceso desde las direcciones IP sospechosas. Sin embargo, este enfoque manual puede ser limitado en su capacidad para adaptarse a nuevos tipos de ataques y puede requerir una supervisión constante para mantener la eficacia.

2. Preparar los Datos:

2.1 Descripción de los Datos:

2.1.1 Describir el alcance de los datos disponibles: Los datos disponibles pueden incluir registros de tráfico de red, registros de eventos de inicio de sesión, direcciones IP, timestamps y acciones realizadas en el sistema (intentos de inicio de sesión, accesos exitosos, etc.).

2.1.2 Describir los datos que no están disponibles pero que son deseables: Sería deseable contar con información adicional sobre los patrones de comportamiento de los usuarios legítimos, como la frecuencia y el horario típico de inicio de sesión, así como datos sobre patrones de ataques de fuerza bruta conocidos.

2.1.3 Describir los datos disponibles que no necesitas: Los datos irrelevantes podrían incluir información sobre actividades no relacionadas con la autenticación de usuarios, registros de sistemas y servicios que no son relevantes para la detección de ataques de fuerza bruta, o registros de tráfico de red de aplicaciones y servicios externos que no representan una amenaza directa.

2.2 Procesamiento de Datos:

2.2.1 Formatear los datos para que estén en una forma con la que puedas trabajar: Los datos deben ser estructurados de manera que sean compatibles con las herramientas y algoritmos de machine learning. Esto puede implicar la conversión de datos de formato de registro a formato tabular, donde cada fila representa una instancia y cada columna representa un atributo.

2.2.2 Limpiar los datos para que sean uniformes y consistentes:

- Imputar valores faltantes: Rellenar o eliminar registros con valores faltantes para garantizar que los datos estén completos.
- Si los valores faltantes son pocos y se pueden inferir de manera confiable, se pueden imputar utilizando técnicas como la media, la mediana o el valor más frecuente.
- Si la cantidad de valores faltantes es significativa o no se pueden inferir de manera confiable, es preferible eliminar esos registros para evitar sesgar el análisis.
- • Identificar y eliminar valores atípicos: Detectar y eliminar registros que contengan valores extremos que puedan sesgar el análisis.
- Utiliza métodos estadísticos como el rango intercuartílico (IQR) o el z-score para identificar valores atípicos.
- Elimina los valores atípicos que estén claramente fuera del rango esperado y que puedan afectar la calidad de los resultados.

2.2.3 Muestrear los datos para encontrar el mejor equilibrio entre redundancia y fidelidad:

- Muestrear instancias: Seleccionar una muestra representativa de los datos para reducir el tamaño del conjunto de datos.
- Utiliza métodos de muestreo aleatorio para seleccionar una muestra que preserve la distribución de los datos originales y sea representativa de la población subyacente.
- Ajusta el tamaño de la muestra según los recursos disponibles y la necesidad de reducir el tiempo de procesamiento sin comprometer la calidad del análisis.
- Reequilibrar clases: Si hay un desequilibrio en las clases objetivo (por ejemplo, más intentos de inicio de sesión legítimos que intentos de inicio de sesión maliciosos), se puede aplicar muestreo para equilibrarlas.
- Utiliza técnicas de muestreo como el muestreo aleatorio estratificado o el muestreo por subconjuntos para obtener una proporción equilibrada de ejemplos de cada clase.
- Asegúrate de mantener la proporción relativa de ejemplos de cada clase en el conjunto de datos para evitar sesgar el modelo hacia la clase mayoritaria.

2.3 Transformación de Datos:

2.3.1 Crear transformaciones lineales y no lineales de todos los atributos:

- El objetivo es optimizar la capacidad del modelo resultante para capturar la información relevante y mejorar su rendimiento predictivo.
- Realizar transformaciones lineales y no lineales en todos los atributos para mejorar la capacidad de los algoritmos de machine learning para descubrir patrones en los datos.
- Las transformaciones lineales como la estandarización y la normalización son útiles para asegurar que todas las características estén en la misma escala, lo que puede mejorar el rendimiento de algoritmos sensibles a la escala, como la regresión logística o las SVM.

- Las transformaciones no lineales, como elevar al cuadrado o tomar la raíz cuadrada de las características, pueden ayudar a capturar relaciones complejas entre las características y mejorar la capacidad del modelo para capturar patrones no lineales en los datos. Esto puede implicar operaciones como elevar al cuadrado, obtener la raíz cuadrada, estandarizar y normalizar los atributos según sea necesario.
- Elevar al cuadrado o tomar la raíz cuadrada de características puede ayudar a capturar relaciones no lineales y mejorar la separabilidad entre clases.
- La estandarización y la normalización son útiles para garantizar que las características tengan una distribución comparable, lo que puede mejorar el rendimiento de algoritmos sensibles a la escala.
- La discretización de características numéricas puede convertirlas en variables categóricas, lo que puede ser útil para ciertos algoritmos o para capturar relaciones no lineales de manera más efectiva.
- El objetivo es optimizar la capacidad del modelo resultante para capturar la información relevante y mejorar su rendimiento predictivo.
- Al aplicar estas transformaciones, se busca mejorar la capacidad del modelo para distinguir patrones relevantes en los datos y mejorar su capacidad predictiva en la detección de ataques de fuerza bruta.

2.3.2 Descomponer atributos complejos en sus partes constituyentes:

- Por ejemplo, si hay atributos que representan combinaciones de características, como la longitud y latitud para la ubicación IP, podrían descomponerse en atributos separados para facilitar el análisis.
- La descomposición de atributos complejos en características individuales puede ayudar a reducir la dimensionalidad del conjunto de datos y a capturar de manera más efectiva las relaciones entre las características.
- Por ejemplo, en el caso de la ubicación IP, la descomposición de la longitud y la latitud en características separadas puede ayudar al modelo a distinguir mejor entre ubicaciones geográficas y detectar patrones específicos asociados con ciertas regiones o países.

3. Evaluar Algoritmos:

3.1 Crear un Conjunto de Pruebas:

Crear un conjunto de validación para su posterior uso:

- Dividimos el conjunto de datos en conjuntos de entrenamiento y prueba, asegurando que contenga ejemplos tanto de intentos legítimos de inicio de sesión como de ataques de fuerza bruta.
- Dado que los ataques de fuerza bruta pueden ser raros en comparación con los intentos de inicio de sesión legítimos, consideramos técnicas como el muestreo proporcional para asegurarnos de que el conjunto de prueba sea representativo.

Seleccionar una medida de rendimiento utilizada para evaluar los modelos:

- Dado que el objetivo principal es detectar ataques de fuerza bruta, las métricas de rendimiento relevantes incluyen la precisión, la sensibilidad (tasa de verdaderos positivos) y la especificidad (tasa de verdaderos negativos).

3.2 Evaluar Algoritmos Candidatos:

Seleccionar un conjunto diverso de algoritmos para evaluar:

- Algoritmos como Random Forest, Support Vector Machines (SVM), Redes Neuronales Artificiales (ANN) y Gradient Boosting Machines (GBM) son candidatos adecuados para la detección de ataques de fuerza bruta.
- También podemos considerar enfoques específicos para la detección de anomalías, como One-Class SVM o métodos basados en clustering.

Utilizar configuraciones de parámetros de algoritmos comunes o estándar:

- Experimentar con diferentes configuraciones de parámetros para cada algoritmo, teniendo en cuenta características específicas de detección de ataques de fuerza bruta, como el número máximo de intentos de inicio de sesión permitidos antes del bloqueo.
- Podemos realizar ajustes finos utilizando técnicas de optimización como la búsqueda en cuadrícula o la optimización bayesiana.
- Al evaluar los algoritmos, buscamos aquellos que maximicen la detección de ataques de fuerza bruta mientras minimizamos los falsos positivos, ya que un exceso de bloqueos de usuarios legítimos puede ser perjudicial para la experiencia del usuario. Una vez que hayamos evaluado y comparado los algoritmos, podremos seleccionar el que mejor se ajuste a nuestras necesidades de protección contra ataques de fuerza bruta.

4. Mejorar los Resultados:

4.1 Ajuste del Algoritmo:

Utilizar parámetros de modelo históricamente efectivos:

- Examinamos la literatura y casos de estudio relacionados con la detección de ataques de fuerza bruta para identificar los parámetros que han demostrado ser efectivos en algoritmos similares.
- Por ejemplo, en un algoritmo como Random Forest, parámetros como el número de árboles en el bosque, la profundidad máxima de cada árbol y el número mínimo de muestras requeridas para dividir un nodo son críticos y pueden ajustarse para mejorar el rendimiento del modelo.

Explorar el espacio de parámetros del modelo:

- Empleamos técnicas como la búsqueda en cuadrícula o la búsqueda aleatoria para explorar diferentes combinaciones de parámetros.

- Por ejemplo, en una SVM, ajustamos parámetros como el tipo de kernel, el coeficiente de regularización (C) y el parámetro de gamma para encontrar la combinación óptima que maximice el rendimiento del modelo en la detección de ataques de fuerza bruta.

Optimizar los parámetros del modelo con buen rendimiento:

- Una vez identificado un conjunto inicial de parámetros prometedores, utilizamos métodos más avanzados de optimización, como la optimización bayesiana, para encontrar la combinación óptima de parámetros.
- Estos métodos son especialmente útiles cuando el espacio de búsqueda de parámetros es grande y complejo.

4.2 Métodos de Conjunto:

Utilizar Bagging en modelos con buen rendimiento:

- Aplicamos la técnica de bagging para entrenar múltiples instancias del mismo algoritmo con diferentes subconjuntos de datos de entrenamiento.
- Por ejemplo, entrenamos múltiples instancias de Random Forest con diferentes muestras de datos de entrenamiento y promediamos sus predicciones para reducir la varianza del modelo y mejorar su estabilidad.

Utilizar Boosting en modelos con buen rendimiento:

- Empleamos la técnica de boosting para entrenar una secuencia de modelos débiles, donde cada modelo se centra en los errores cometidos por el anterior.
- Por ejemplo, utilizamos un algoritmo de boosting como AdaBoost o Gradient Boosting para mejorar gradualmente la capacidad predictiva del modelo en la detección de ataques de fuerza bruta.

Mezclar los resultados de modelos con buen rendimiento:

- Consideramos técnicas de ensamblaje como la votación o el stacking para combinar las predicciones de múltiples modelos con buen rendimiento.
- Por ejemplo, podemos utilizar un enfoque de votación mayoritaria donde cada modelo tiene un voto ponderado según su rendimiento en el conjunto de validación.

4.3 Selección del Modelo:

Seleccionar un subconjunto diverso de modelos con buen rendimiento (5-10):

- Para asegurar una cobertura amplia y efectiva, evaluamos una variedad de modelos con buen rendimiento en un conjunto de validación. Buscamos modelos que complementen y diversifiquen las fortalezas y debilidades de cada uno. Por ejemplo, podríamos elegir una combinación que incluya modelos lineales y no lineales, modelos basados en árboles y modelos de ensamblaje. Esta diversificación nos permite abordar una amplia gama de posibles escenarios de detección de ataques de fuerza bruta, maximizando así nuestra capacidad para identificar y responder a diversas amenazas.

Evaluar modelos con buen rendimiento en un conjunto de validación:

- Utilizamos un conjunto de validación separado para evaluar el rendimiento de los modelos seleccionados y garantizar su capacidad de generalización a datos no vistos. Durante esta evaluación, ajustamos los parámetros del modelo según sea necesario para optimizar su rendimiento en la tarea específica de protección contra ataques de fuerza bruta. Validamos nuevamente el rendimiento de los modelos ajustados para garantizar que estén correctamente optimizados y listos para su implementación en un entorno de producción.

Seleccionar un pequeño grupo de modelos con buen rendimiento (1-3):

- Finalmente, después de una exhaustiva evaluación y ajuste, elegimos un pequeño conjunto de modelos con el mejor rendimiento en el conjunto de validación. Es esencial equilibrar la complejidad del modelo con su capacidad para generalizar y adaptarse a nuevas amenazas. Por lo tanto, seleccionamos los modelos que ofrecen el mejor equilibrio entre rendimiento y eficiencia, garantizando así una protección efectiva contra los ataques de fuerza bruta sin comprometer la operatividad del sistema.