בס"ד

**check phishing sms**



מאת:

איתמר אביר ועידו אלמקייס

**הקדמה**

**רקע כללי**

בעידן הדיגיטלי המתקדם, הודעות דיוג (Smishing) הפכו לאחד האיומים הנפוצים והמסוכנים ביותר על משתמשים פרטיים ועסקיים כאחד. התקפות אלו, המשתמשות בהודעות SMS כדי להתחזות לגופים לגיטימיים כמו בנקים, חברות שליחויות, או שירותי ממשלה, מנצלות פגיעות אנושית כדי לגנוב מידע אישי, סיסמאות או פרטים פיננסיים. התחכום הגובר של הודעות אלו הופך את הזיהוי שלהן למשימה מאתגרת עבור המשתמש הממוצע, ולכן קיים צורך הולך וגובר בפתרונות טכנולוגיים שיכולים לסייע בהגנה מפניהן.

**הפתרון המוצע**

פרויקט זה מציע פתרון טכנולוגי מקיף לבעיה זו, באמצעות יישום אפליקציית web שלמה המשלבת למידת מכונה עם פיתוח אינטרנט.

האפליקציה מאפשרת למשתמש להזין הודעת טקסט שקיבל ולשלוח אותה לבדיקה. בקשת המשתמש נשלחת לשרת, שם מודל למידת מכונה שאומן על מאגר נתונים רלוונטי מנתח את ההודעה. המודל מסוגל לסווג את ההודעה כהודעת דיוג או כהודעה לגיטימית. אם ההודעה מסווגת כחשודה, המערכת מזהה ומשתפת את המילים המרכזיות בהודעה שהובילו לסיווג זה.

לאחר שהבדיקה מסתיימת, תוצאת הסיווג נשלחת בחזרה למשתמש ומוצגת בצורה ברורה בממשק האפליקציה. המשתמש מקבל אינדיקציה האם ההודעה "אמינה" או "הודעת דיוג". במקרה של הודעת דיוג, יוצג בנוסף פירוט של המילים החשודות שנמצאו בהודעה. המערכת נועדה להיות קלה ונוחה לשימוש, ומספקת הגנה בזמן אמת מפני איומי סייבר נפוצים.

**רקע תשתיתי**

האפליקציה פותחה על בסיס ארכיטקטורה מודולרית המורכבת מצד לקוח וצד שרת, במטרה ליצור מערכת יעילה, ניתנת להרחבה וקלה לתחזוקה.

1. **צד הלקוח** - האפליקציה בצד הלקוח נבנתה באמצעות ספריית **React**, בשפת JavaScript פופולרית לפיתוח ממשקי משתמש. הבחירה ב-React אפשרה לבנות ממשק משתמש דינמי המגיב בזמן אמת לפעולות המשתמש ולנתונים המתקבלים מהשרת.

**טכנולוגיות בצד הלקוח:**

* + **React Hooks**: שימוש ב-React Hooks, ובפרט useState, מאפשר לנהל את מצב האפליקציה, כמו תוכן ההודעה, תוצאת הסיווג ומצב טעינה.
  + **API של fetch**: הממשק משתמש ב-API של fetch כדי לשלוח בקשות POST לשרת ולקבל בחזרה את התוצאות בפורמט JSON.

1. **צד השרת** - צד השרת כתוב כולו בשפת **Python**. בחירה זו נבעה מיכולותיה המצוינות של Python בתחום למידת המכונה ועיבוד נתונים. השרת מופעל באמצעות **Flask**, מיקרו-פריימוורק אינטרנטי קל משקל.

**טכנולוגיות בצד השרת:**

* + **Flask**: פריימוורק שאיפשר יצירת נקודת קצה (endpoint) ייעודית לקבלת בקשות מהממשק הקדמי בצורה יעילה וקלת משקל.
  + **Pandas**: ספרייה המשמשת לטעינה, ניתוח וניהול בסיס הנתונים הקיים, המאוחסן בקובץ smishingDB\_augmented.csv.
  + **sklearn.model\_selection**: מאפשרת לחלק את הנתונים לקבוצת אימון וקבוצת בדיקה.
  + **TfidfVectorizer**: ממירה הודעות טקסט לווקטורים מספריים שניתנים לעיבוד על ידי המודל.
  + **LinearSVC, RandomForestClassifier, XGBoost**: מודלים למידת מכונה המשמש לסיווג ההודעות.
  + **flask-cors**: מותקנת על מנת לאפשר לממשק הקדמי, הפועל על פורט 3000, לתקשר עם השרת הפועל על פורט 8000.

**המצב הקיים**

ישנם מוצרים שונים בשוק שמספקים פתרונות דומים להגנה מפני Smishing. באופן כללי, רוב הפתרונות הללו מתחלקים למספר קבוצות:

* **שמירת רשימות שחורות (BlackListing)**: בין אם בצד השרת או בצד הלקוח, מתבצעת שמירה של רשימות של הודעות או מספרי טלפון הידועים כחשודים.
* **שימוש בלמידת מכונה**: נעשה ניסיון לסווג הודעות לפישינג או לא באמצעות מודלים שאומנו על דאטה רחב.
* **שימוש באלגוריתמים**: אלגוריתמים המחפשים מילות מפתח, דפוסים וביטויים חשודים בהודעות.

נראה מספר אפליקציות עיקריות המוכרות בתחום:

* **TrendMicro Check**: אפליקציה מבית TrendMicro שמשמשת כמעין אנטי-וירוס שבין היתר בודק הודעות. היא משתמשת בטכנולוגיית AI לסיווג ההודעות ומונעת ממשתמשים לראות הודעות המסווגות כמסוכנות.
* **SpamHound**: אפליקציה שמשתמשת ברשימת חוקים (מילות מפתח, רשימות שחורות) בכדי לסנן הודעות דיוג.
* **RoboKiller**: אפליקציה שמשתמשת בלמידת מכונה בכדי לנתח תוכן הודעות. היא משתמשת באלגוריתמים שמזהים דפוסים ומילות מפתח חשודות בהודעות.

**הערך המוסף של check phishing sms**

אפליקציית *check phishing* *sms*מציענ גישה משולבת ויעילה יותר לזיהוי והגנה מפני הודעות דיוג, בכך שהיא משלבת בין הגישות השונות ומנתחת כל הודעה ממספר כיוונים בו זמנית בשביל להשיג את הסיווג הטוב ביותר. שאר האפליקציות מבוססות על רשימות שחורות, חוקים/מילות מפתח, או מודלים יחידניים. הגישה שלנו משלבת שלושה מודלים משלימים יחד עם "הצבעה" לשיפור יציבות ודיוק, לצד *שקיפות* מוסברת דרך המילים החשודות.

* **שילוב טכנולוגיות מתקדמות**: המערכת משתמשת בשלושה מודלים למידת מכונה שונות כדי לקבל סיווג ע"פ כמה שיטות (LinearSVC, RandomForestClassifier, XGBoost) שמאומנים על בסיס נתונים קיים של הודעות דיוג והודעות לגיטימיות, כדי לסווג הודעות חדשות בצורה מדויקת.
* **"הצבעה"**: ע"פ שלושת המודלים נקבע האם ההודעה חשודה או לא ומסווגת ככזאת. אם 2 מתוך 3 המודלים מראים כהודעה חשודה. ההודעה תתווסף עם התווית המתאימה אחרת עם התווית התואמת כהודעה לא חשודה.
* **מנגנון זיהוי מילים חשודות**: בנוסף לסיווג הכולל, המערכת מזהה ומציגה למשתמש את חמש המילים החשודות ביותר בהודעה, ובכך מספקת שקיפות ומאפשרת למשתמש להבין טוב יותר מדוע ההודעה סווגה כחשודה.
* **למידה מתמשכת**: המערכת כוללת מנגנון ייחודי של למידה עצמית. כל הודעה חדשה שנבדקה מסווגת ונשמרת בבסיס הנתונים הקיים (smishingDB\_augmented.csv), מה שמאפשר למודל למידת המכונה להשתפר ולהיות מדויק יותר עם הזמן. גישה זו מבטיחה שהאפליקציה תישאר רלוונטית ויעילה מול התקפות מתפתחות.

**אפיון ועיצוב המערכת**

**מטרות המערכת**

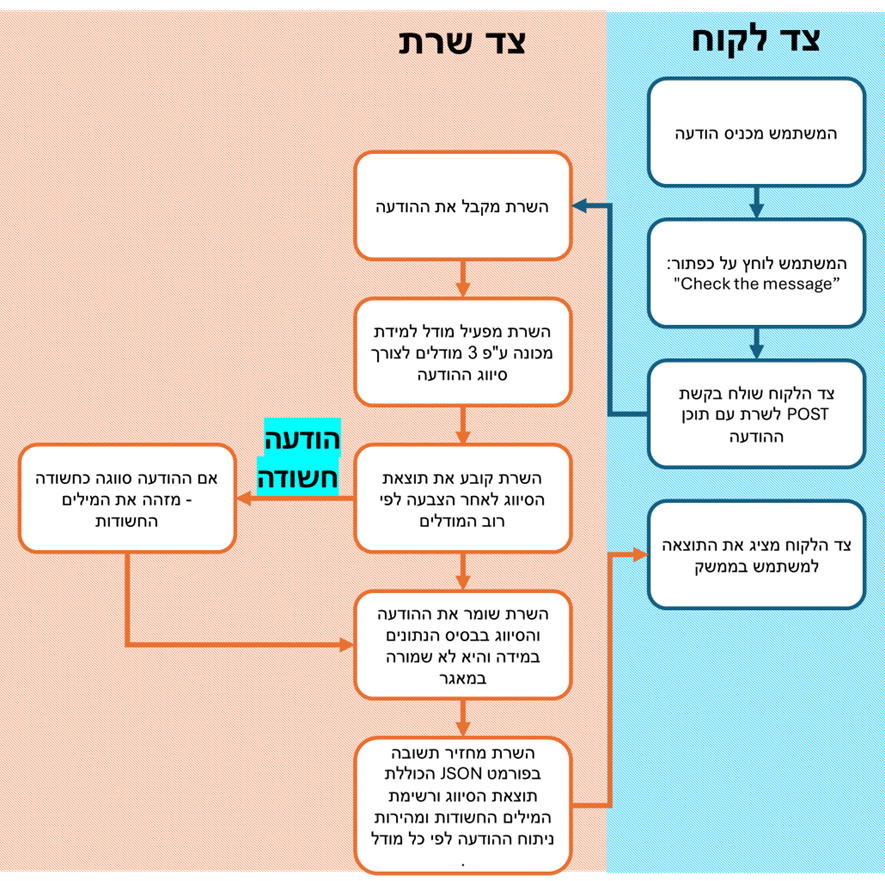
המטרה העיקרית של המערכת היא לספק למשתמש כלי יעיל ונוח לזיהוי הודעות דיוג (Smishing) והגנה מפניהן. המערכת שואפת להשיג זאת באמצעות מספר מטרות משנה:

* **סיווג מדויק**: לסווג הודעות טקסט כ"הודעת דיוג" או "הודעה לגיטימית" באמצעות מודל למידת מכונה.
* **הגברת מודעות**: לספק למשתמש הסבר ברור מדוע הודעה מסוימת סווגה כחשודה, על ידי הצגת רשימת המילים החשודות ביותר.
* **למידה מתמשכת**: לאפשר למודל למידת המכונה להשתפר עם הזמן על ידי שמירת הודעות חדשות שסווגו בבסיס הנתונים.
* **חווית משתמש פשוטה**: להציע ממשק משתמש נקי וקל לשימוש, המאפשר בדיקה מהירה של הודעות.

**העיצוב הכללי של המערכת**

המערכת מורכבת משני רכיבים עיקריים שפועלים באופן סינכרוני: צד לקוח (Frontend) וצד שרת (Backend). הפרדת האחריות בין הרכיבים מאפשרת גמישות בפיתוח ותחזוקה.

* **צד הלקוח (Frontend)**:
  + **ממשק משתמש**: נבנה באמצעות ספריית React ומעוצב בפשטות. הממשק כולל תיבת טקסט גדולה בה המשתמש יכול להזין את תוכן ההודעה החשודה.
  + **תקשורת**: הממשק שולח בקשת POST לשרת, המכילה את ההודעה לבדיקה. בזמן ההמתנה, מוצג מצב טעינה למשתמש.
  + **הצגת תוצאות**: לאחר קבלת התגובה מהשרת, הממשק מציג בצורה ברורה את תוצאת הסיווג (הודעה לגיטימית או הודעת דיוג לפי כל אחד מהמודלים). אם ההודעה סווגה כחשודה, מוצגת גם רשימת מילים חשודות שהובילו לסיווג זה. במידה וההודעה כבר קיימת במאגר, תוצג הודעה בהתאם.
* **צד השרת (Backend)**:
  + **ארכיטקטורה**: השרת, שנבנה ב-Python עם פריימוורק Flask, מקבל את בקשת הבדיקה מצד הלקוח.
  + **מודל למידת מכונה**: השרת מפעיל את המודל TfidfVectorizer משתמש בו לצורף המרת טקסט לווקטורים מספריים. (כי מערכת מחשב מבינה רק מספרים) לאחר מכן ההודעה עוברת דרך 3 מודלים כדי לנתח את ההודעה.
  + **סיווג ההודעה**: לאחר הניתוח ההודעה תסווג בהתאם לבחירת לפחות 2 מתוך 3 המודלים.
  + **שמירה ושיפור**: השרת מוסיף את ההודעה החדשה ואת סיווגה לקובץ ה-smishingDB\_augmented.csv כדי לעדכן את בסיס הנתונים ולאפשר למידה מתמשכת.
  + **החזרת נתונים**: השרת מחזיר לממשק הקדמי תשובה בפורמט JSON, הכוללת את תוצאת הסיווג ואת המילים החשודות שנמצאו.

**תרשים זרימת עבודה**

**היישום**

**צד הלקוח**

צד הלקוח נבנה באמצעות ספריית **React**, המאפשרת יצירת ממשק משתמש אינטואיטיבי ודינמי. הממשק מיועד לנהל את התצוגה, אינטראקציית המשתמש והתקשורת מול השרת.

1. **הפעלה וניטור הודעות**

כאשר המשתמש מפעיל את האפליקציה, הוא מזין את ההודעה הרצויה בשדה טקסט. הנתונים מנוטרים בזמן אמת באמצעות React Hooks, ומאפשרים שליטה מלאה על מצב האפליקציה. האפליקציה ממתינה לפעולת המשתמש כדי לשלוח את ההודעה לשרת.

1. **תצוגת התוצאות**

התצוגה של התוצאות מתבססת על הנתונים המתקבלים מהשרת. כאשר התשובה מתקבלת, הממשק מציג אותה באופן מיידי. אם התוצאה היא "הודעת דיוג", מוצגות גם המילים החשודות שנמצאו על ידי השרת.

1. **תקשורת**

התקשורת עם השרת מתבצעת בשלב אחד בלבד:

* + כאשר המשתמש לוחץ על כפתור השליחה, האפליקציה שולחת בקשת POST ישירות לשרת דרך ה-API של fetch. בקשה זו מכילה את תוכן ההודעה.
  + השרת מעבד את הבקשה ומחזיר למשתמש תשובה בפורמט JSON, המכילה את תוצאת הסיווג ואת רשימת המילים החשודות.

**צד השרת**

צד השרת הוא ליבת המערכת ואחראי על קבלת בקשות ממשתמשים, ניתוח ההודעות והחזרת תשובות. הוא מורכב משני מודולים עיקריים: מודול התקשורת ומודול ניתוח ההודעות. בנוסף, קיים מודול תומך שאחראי על יצירת ואימון המודלים הרלוונטיים.

1. **מודול התקשורת**

מודול התקשורת אחראי לקבל את הבקשות מהמשתמשים ולהחזיר תשובות. השרת פועל על פלטפורמת Flask, ומציג נקודת קצה (/check) אליה מתקבלות הבקשות. כאשר מתקבלת בקשה, מודול זה מעביר את ההודעה למודול הניתוח וממתין לתשובה.

1. **מודול ניתוח ההודעה**

מודול זה אחראי על הפעלת מודל למידת המכונה. הוא מקבל את ההודעה, מפעיל עליה את המודל שנוצר באמצעות TfidfVectorizer ושלושת המודלים (LinearSVC, RandomForestClassifier, XGBoost) , ומבצעים סיווג. לאחר מכן, הוא מזהה את המילים החשודות ביותר ומחזיר את כל הנתונים הרלוונטיים למודול התקשורת. בנוסף, הנתונים החדשים נשמרים בקובץ ה-smishingDB\_augmented.csv לטובת למידה מתמשכת.

1. **בסיס הנתונים**

בסיס הנתונים מנוהל באמצעות קובץ CSV פשוט. קובץ זה משמש את המודל לאימון ולאחסון הודעות חדשות, ומאפשר למערכת ללמוד ולהשתפר לאורך זמן.

בניגוד למערכת מורכבת עם טבלאות מרובות וכלים כמו MySQL, בסיס הנתונים הנוכחי הינו קובץ CSV, ולכן אין תהליכים כמו מחיקת רשומות על בסיס זמן או שמירת צילומי מסך. הנתונים היחידים שנשמרים הם תוכן ההודעה והסיווג שלה.

**מודול ניתוח ההודעה**

מודול ניתוח ההודעה הוא הלב הפועם של המערכת, והוא אחראי על עיבוד, ניתוח וסיווג ההודעות. מודול זה נבנה על בסיס טכנולוגיות של למידת מכונה ועיבוד שפה טבעית (NLP).

**המודל**

המודל מורכב משני שלבים עיקריים:

**וקטוריזציה (Vectorization):** השלב הראשון הוא המרת הודעת הטקסט שמתקבלת לווקטור מספרי. טקסט אינו יכול להיות מעובד ישירות על ידי מודלים של למידת מכונה, ולכן יש צורך להפוך אותו לייצוג מספרי. לשם כך, אנו משתמשים ב-TfidfVectorizer (Term Frequency-Inverse Document Frequency). כלי זה בודק את תדירות הופעתה של מילה בהודעה (TF) ואת חשיבותה ביחס לכלל בסיס הנתונים (IDF). התוצאה היא וקטור של מספרים המייצג את ההודעה באופן כזה, שמילים חשובות וייחודיות מקבלות משקל גבוה יותר.

**סיווג (Classification):** השלב השני הוא שימוש במודל למידת מכונה כדי לסווג את הווקטור שנוצר כ"הודעת דיוג" או "הודעה לגיטימית". לשם כך, אנו משתמשים במודלים שונים כדי לקבל דיוק ובדיקה ע"פ מס' שיטות שונות. מודל זה הוא למעשה "יער" של עצי החלטה, שכל אחד מהם מקבל החלטה עצמאית. המודל מסווג את ההודעה על בסיס רוב ההחלטות של העצים שבו. מודלים נבחרו בגלל יכולתם לטפל במערכי נתונים גדולים, יציבותם ורמת הדיוק הגבוהה שלהם.

**תהליך העבודה של המודול**

* + קבלת בקשה: המודול מקבל את תוכן ההודעה ממודול התקשורת.
  + ניתוח וסיווג: ההודעה מוזנת לתוך המודל. המודל מבצע תחילה וקטוריזציה של ההודעה, ולאחר מכן משתמש במודל ה-RandomForestClassifier כדי לבצע סיווג.
  + זיהוי מילים חשודות: אם ההודעה סווגה כ"הודעת דיוג" (תוצאה של 1), המודול מפעיל פונקציה נוספת שמטרתה לזהות את המילים החשודות ביותר. הדבר נעשה על ידי ניתוח ציוני ה-TF-IDF של המילים בהודעה. המילים בעלות הציון הגבוה ביותר, אלו שנחשבות הכי ייחודיות וחשובות, מזוהות כחשודות.
  + למידה מתמשכת: לאחר הניתוח, ההודעה המקורית ותוצאת הסיווג נשמרות בקובץ ה-CSV המהווה את בסיס הנתונים של המערכת. תהליך זה של למידה מתמשכת מאפשר למודל לשפר את דיוקו עם כל הודעה חדשה שהוא פוגש.
  + החזרת התוצאות: לבסוף, מודול הניתוח מחזיר למודול התקשורת את תוצאת הסיווג (1 או 0) ואת רשימת המילים החשודות שנמצאו, אם רלוונטי.

**המודלים והאלגוריתמים בקוד**

תהליך העיבוד והחיזוי במערכת כולל שלושה שלבים עיקריים:

המרה לוקטורים באמצעות TfidfVectorizer, חיזוי בעזרת שלושה מסווגים לבסוף הצבעה ע"פ המסווגים כדי לקבוע האם ההודעה פישינג.

* 1. **TfidfVectorizer - וקטוריזציה של טקסט**

זהו מודל שמגיע מספריית scikit-learn ואחראי על המרת הטקסט הגולמי של ההודעה לפורמט מספרי שניתן לעיבוד על ידי מודל למידת המכונה. האלגוריתם פועל על בסיס חישוב TF-IDF (Term Frequency-Inverse Document Frequency):

**Term Frequency (TF):** בודק כמה פעמים מילה מסוימת מופיעה בהודעה נתונה. ככל שמילה מופיעה יותר, כך היא נחשבת חשובה יותר בתוך ההודעה עצמה.

**Inverse Document Frequency (IDF):** בודק כמה פעמים מילה מסוימת מופיעה בכלל ההודעות בבסיס הנתונים. מילים נפוצות מאוד (כמו "של", "אני", "את") מקבלות ציון נמוך, בעוד שמילים נדירות וייחודיות מקבלות ציון גבוה.

השילוב של שני המדדים הללו יוצר וקטור מספרי שמייצג את ההודעה. מילים שמקבלות ציון TF-IDF גבוה הן מילים ייחודיות ורלוונטיות, והן משמשות את המודל לזהות דפוסים חשודים.

1. **Ensemble Learning - שילוב מסווגים להודעות**

ליבת המערכת בנויה על שלושה מודלים של למידת מכונה הפועלים יחדיו במבנה של Ensemble Learning – שילוב בין מסווגים שונים לקבלת החלטה מדויקת ויציבה יותר.

הגישה הזו מאפשרת למערכת להתמודד עם סוגים שונים של הודעות טקסט ולשפר את אחוז הדיוק לעומת שימוש במודל יחיד בלבד.

1. **RandomForestClassifier - סיווג הודעות**

לאחר שהטקסט הומר לווקטור מספרי, הוא עובר למודל RandomForestClassifier. זהו מודל סיווג מתקדם, שמגיע גם הוא מספריית scikit-learn, ופועל על העקרונות הבאים:

אנסמבל של עצי החלטה: המודל מורכב מ"יער" (Forest) של עצי החלטה (Decision Trees). כל עץ החלטה הוא מודל פשוט שמבצע סדרה של החלטות על הנתונים (במקרה שלנו, על וקטור ה-TF-IDF של ההודעה).

החלטה משותפת: כדי לקבל את הסיווג הסופי, המודל משתמש בהצבעה (Voting) של כל עצי ההחלטה. אם רוב העצים קובעים שההודעה היא "דיוג", זו תהיה התוצאה הסופית.

השימוש ב"יער" במקום בעץ יחיד הופך את המודל לחזק ועמיד יותר בפני נתוני אימון לא מושלמים. הוא מסוגל לזהות דפוסים מורכבים בנתונים ולבצע סיווג מדויק.

1. **LinearSVC – מסווג וקטורים קווי**

מודל נוסף אליו עוברים הנתונים הוא מודל LinearSVC, היחודיות שלו שהוא מבוסס על אלגוריתם Support Vector Machine.

מודל זה מנסה למצוא "קו גבול" (Hyperplane) שמפריד בצורה מיטבית בין שתי הקבוצות: הודעות דיוג (1) והודעות לגיטימיות (0).

היתרונות של LinearSVC הם:

* יעילות ומהירות גבוהה – מתאים במיוחד למידע טקסטואלי רחב.
* ביצועים טובים גם כאשר יש הרבה מאפיינים (Features) – מצב שכיח בטקסטים.
* עמידות בפני Overfitting בזכות בחירה חכמה של גבול ההפרדה.

LinearSVC הוא למעשה המודל הקל והמהיר ביותר מבין השלושה, והוא מספק "קול" מהיר להצבעה, לעיתים גם במקרים שבהם שני המודלים האחרים מתלבטים.

1. **XGBClassifier – גרדיאנט בוסטינג חכם**

המודל השלישי הוא XGBoost, אחד האלגוריתמים הפופולריים ביותר בלמידת מכונה מודרנית.

שלא כמו RandomForest, שבו כל העצים פועלים במקביל, XGBoost בונה את העצים באופן סדרתי: כל עץ חדש לומד מהטעויות של הקודמים לו.

כך המודל לומד לתקן את עצמו בהדרגה, כאשר כל עץ ממוקד באזורים שבהם המודל הקודם טעה. התוצאה היא סיווג חזק מאוד שמגיע לרמות דיוק גבוהות במיוחד גם בסטים קטנים יחסית של נתונים.

יתרונותיו של XGBoost:

* ביצועים גבוהים – נחשב אחד המודלים המדויקים ביותר בתחום הסיווג.
* אופטימיזציה חכמה – מנצל את משאבי המחשב ביעילות ומאפשר חיזוי מהיר.
* יכולת הכללה טובה – מתאים למידע חדש, גם אם הוא שונה חלקית מהדאטה המקורי.

לאחר שכל אחד משלושת המודלים מחזיר את התחזית שלו (0 או 1), המערכת מבצעת הצבעה. אם לפחות שניים מתוך שלושת המודלים סיווגו את ההודעה כחשודה (1), התוצאה הסופית תוגדר כהודעת דיוג.

שיטה זו מאזנת בין מהירות (LinearSVC) לדיוק (XGBoost) ויציבות (RandomForest), ומספקת סיווג אמין גם במקרים גבוליים.

בנוסף, המערכת מחשבת ומחזירה ללקוח את זמן החיזוי של כל מודל, כדי לאפשר ניתוח ביצועים והשוואה בין המסווגים.

1. **זיהוי מילים חשודות**

לאחר הסיווג ההודעה כחשודה המערכת מפעילה אלגוריתם לזיהוי עד 5 מילים חשודות אשר נשמרות בפורמט JSON לצורך החזרה לצד הלקוח כדי שהמשתמש ילמד ויזהה מילים חשודות בעצמו

**נתונים ותוצאות**

**מקור הנתונים**

הפרויקט מתבסס על **מערך נתונים קיים ורלוונטי**, וכל תהליכי העיבוד והאלגוריתמים בו מכוונים להפקת תוצאות מדויקות ואמינות ככל האפשר.  
המודלים אומנו על קובץ הנתונים smishingDB\_augmented.csv, שמשמש כבסיס הלמידה של המערכת.

קובץ זה מכיל שתי עמודות עיקריות:

**TEXT** –תוכן הודעת SMSכפי שהיא נשלחה.

**LABEL** – סיווג ההודעה:

ספרה 1 מייצגת הודעת Phishing

ספרה 2 מייצגת הודעהSafe

**דוגמאות מתוך בסיס הנתונים**

| **TEXT** | **LABEL** |
| --- | --- |
| “Congratulations! You've come through group A free gift. Claim it here: http://freegift.com/claim.” | 1 |
| “Reminder: Your telephone service appointment is scheduled on 4/19/2024.” | 0 |
| “Get a free consultation with our pet nutritionist. Book now at: http://tinyurl.com/791252” | 1 |

הדוגמאות מייצגות את שני סוגי ההודעות – הודעות דיוג הכוללות לינקים חשודים או ניסוחים מפתים, והודעות לגיטימיות בעלות ניסוח ניטרלי ושגרתי.

**הרחבת הדאטה ולמידה מתמשכת**

אחד המאפיינים המרכזיים במערכת הוא היכולת להוסיף נתונים חדשים בזמן אמת.  
כאשר המשתמש שולח הודעה לבדיקה, השרת בודק אם ההודעה כבר קיימת בקובץ.  
במידה ולא היא מתווספת אוטומטית לקובץ הנתונים (smishingDB\_augmented.csv) יחד עם הסיווג שהתקבל על פי הצבעת המודלים.

במערכת זאת קורים התהליכים הבאים:

* מאגר הנתונים גדל כך מקבלים - גיוון
* המודלים נחשפים למבנים לשוניים חדשים - שיפור
* ניתן לבצע אימון חוזר בעתיד כדי לשפר את הביצועים עם הנתונים שנוספו - דיוק

**מדידת ביצועים ותוצאות חיזוי**

בכל קריאה למערכת, נאספים שלושה נתוני חיזוי - אחד מכל מסווג  
(LinearSVC, RandomForest, XGBoost)  
בנוסף, המערכת מודדת את זמן החיזוי (במילישניות) לכל מודל בנפרד ומחזירה מידע זה ללקוח.

כך ניתן לבצע:

* השוואה בין זמני ביצוע של המודלים.
* בדיקה של עקביות התוצאות (האם שלושתם חוזים את אותה תוצאה)
* הסקת מסקנות על יעילות ודיוק בזמן אמת.

הנתונים שנאספים מאפשרים ניתוח ביצועים טכני – לדוגמה, ניתן לראות שמודל LinearSVC מגיב מהר מאוד, בעוד שXGBoost בדרך כלל מדייק יותר אך דורש יותר זמן עיבוד.

**סיכום**

השילוב בין דאטה מתעדכן, מעקב אחרי זמני חיזוי ויכולת הרחבה בזמן אמת יוצר מערכת לומדת, אמינה ויעילה.  
הודות למבנה זה, הפרויקט מספק תוצאות איכותיות, יציבות ומבוססות נתונים אמיתיים תוך התאמה מתמדת לדפוסי תקשורת חדשים שמאפיינים הודעות דיוג מודרניות.

**תהליך אימון המודל**

תהליך אימון המודל מתבצע בעת עליית השרת (בהפעלה הראשונה של המערכת).  
בשלב זה נטען קובץ הנתונים smishingDB\_augmented.csv, הכולל הודעות טקסט וסיווגיהן, והמודלים מאומנים על בסיסו. כל הנתונים משמשים לאימון על מנת למקסם את רמת הדיוק הראשונית.

במהלך האימון המודלים לומדים לזהות דפוסים ומאפיינים ייחודיים של הודעות דיוג לעומת הודעות לגיטימיות, על סמך מילים, ניסוחים ומבנה ההודעה.

**תוצאות ושיפור המודל**

לאחר סיום שלב האימון, המערכת מוכנה לקבל הודעות חדשות מהמשתמשים ולסווג אותן בזמן אמת.  
כאשר הודעה מתקבלת, היא עוברת את השלבים הבאים:

1. וקטוריזציה המרה לייצוג מספרי באמצעות TF-IDF
2. חיזוי בשלושה מודלים – LinearSVC, RandomForest, XGBoost
3. הצבעה – אם לפחות שניים מתוך שלושת המודלים סיווגו את ההודעה כחשודה, היא תוגדר כהודעת דיוג.

בנוסף, המערכת מודדת את זמן החיזוי של כל מודל בנפרד (במילישניות) ומחזירה את הנתונים הללו יחד עם תוצאת הסיווג.  
כך המשתמש מקבל חיווי מדויק ומהיר לגבי סוג ההודעה ואינדקציה איזו מודל הכי מהיר ונכון

**דוגמה לתוצאות:**

אם המשתמש מזין את ההודעה:

“You have won a free iPhone! To claim your prize, please click this link: http://win.prize.co.uk”

המערכת תחזיר תשובה לדוגמה:

{

"results": {

"LinearSVC": {"prediction": 1, "time": 12.4},

"RandomForest":{"prediction": 1, "time": 35.2},

"XGBoost": {"prediction": 1, "time": 28.7}

},

"final\_prediction": 1,

"suspicious\_words": ["free", "won", "prize", "click", "claim"],

"message\_info": "Message added to dataset."

}

**שיפור מתמשך:** אחת היכולות המרכזיות של המערכת היא **למידה מתמשכת**.  
לאחר שכל הודעה נבדקת, היא והסיווג שלה נשמרים בקובץ ה־CSV (smishingDB\_augmented.csv).  
במידה שההודעה כבר קיימת – היא לא תתווסף שוב, אך הודעות חדשות נצברות עם הזמן ומאפשרות למודל להיחשף לדוגמאות נוספות.

תהליך זה מביא ליתרונות ברורים:

* גידול במאגר הנתונים – שיפור כיסוי השפה והניסוחים.
* שיפור יכולת הכללה – המודלים לומדים לזהות דפוסים חדשים.
* דיוק מתמשך – ניתן לבצע אימון מחדש על בסיס הנתונים המעודכן, וכך לשפר את הביצועים לאורך זמן.

**תשתית המערכת**

פרויקט *check phishing sms* מציג פתרון טכנולוגי מלא ואינטגרטיבי להתמודדות עם איום הודעות הדיוג (Smishing) ההולך וגובר. האפליקציה פותחה כיישום אינטרנטי המורכב מצד שרת (Backend) וצד לקוח (Frontend), הפועלים יחד על מנת לספק למשתמש כלי יעיל, מדויק ונוח לזיהוי הודעות חשודות.

**צד השרת(Backend)**

השרת נבנה בשפת Pythonומופעל באמצעות Flask  
הוא אחראי על עיבוד ההודעה, ניתוחה והפעלת שלושת המודלים.  
במהלך הפעולה השרת:

מבצע וקטוריזציה לטקסט בעזרת TfidfVectorizer לאחר מכן מפעיל את המודלים LinearSVC, RandomForestClassifier XGBClassifierובסוף מבצע הצבעה לקביעה סופית האם ההודעה חשודה.

צד השרת מודד זמני חיזוי לכל מודל ומחזיר את התוצאה בפורמט JSON שכולל גם את המילים החשודות.

קיים מנגנוןCORS שתפקידו התקשרות ישירה מול צד הלקוח.

**צד הלקוח(Frontend)**

הצד המשתמש פותח באמצעות React ומציע ממשק פשוט ונוח שבו המשתמש מזין הודעת SMS לבדיקה.  
לאחר שליחה, המידע נשלח לשרת והתגובה מוצגת בצורה ברורה האם ההודעה בטוחה או חשודה, יחד עם זמני החיזוי של כל מודל.

**הערך המוסף של האפליקציה**

האפליקציה מספקת פתרון טכנולוגי מלא ואינטגרטיבי להתמודדות עם איום הודעות הSmishing

* סיווג מדויק ומהיר – בזכות שימוש בשלושה מודלים משלימים.
* למידה עצמית מתמשכת – המערכת משתפרת ככל שמשתמשים בה יותר.
* שקיפות וניתוח ביצועים – זמני החיזוי מאפשרים להבין את יעילות כל מודל.
* הצגת המילים החשודות – מטרה נוספת של האפליקציה היא ללמד את המשתמש להיזהר מהודעות דיוג ע"י תצוגה של מילים חשודות.
* ממשק משתמש נוח – מתאים לשימוש יומיומי ולבדיקה מהירה של הודעות חשודות.

**סיכום**

פרויקט *check phishing sms* מציג פתרון מעשי, חדשני ומדויק לזיהוי הודעות דיוג.  
השילוב בין Flask React ושלושה אלגוריתמים מתקדמים של למידת מכונה יצר מערכת אמינה ויעילה, המסוגלת לזהות ניסיונות Smishing בזמן אמת.  
הודות למנגנון ההצבעה הרובית וליכולת הלמידה המתמשכת עם שמירת כל הודעה במאגרים, האפליקציה מספקת הגנה חכמה ומתעדכנת כנגד איומי סייבר מודרניים וגם למידת המשתמש.

איתמר אביר

עידו אלמקייס