Predicting Popularity of News Articles

Neriya Zudi, Omer Alali

Lecturer:
Tammar Shrot

SCE - Sami Shamoon College of Engineering

Department of Software Engineering

תקציר

בפרויקט אנו מבקשים לבדוק את היכולת לחזות את הפופולריות של מאמרי חדשות באמצעות מודלים מתחום הבינה המלאכותית.

הניתוח שלנו מתייחס למאגר מידע של 40,000 מאמרי חדשות המסכמים קבוצה הטרוגנית של תכונות על מאמרים שפורסמו על ידי Mashable בתקופה של שנתיים, ואנו משווים בין שני מודלים של למידת מכונה (SVM, Naive Bayes) כדי לנתח אותם.

מהתוצאות שערכנו עולה כי מודל Naive Bayes השיג אחוזי דיוק גבוהים יותר בניבוי הפופולאריות, וכן נמצא מועיל יותר במספר מישורים משמעותיים בניהם: עלות מתכנת, סיבוכיות זמן ומקום, ויכולת הרחבה.

סקירת הספרות

חיזוי הפופולריות של כתבות חדשותיות מקוונות הוא חלק מתחום נרחב של מחקר של בעיית חיזוי הפופולריות של תוכן מקוון (למשל ציוצים, פוסטים, סרטונים) עליו נכתבו מחקרים רבים.

פופולריות נמדדת לרוב על ידי התחשבות במספר אינטראקציות ברשת וברשתות החברתיות (למשל, מספר השיתופים, הלייקים והתגובות). חיזוי פופולריות כזו הוא בעל ערך עבור מחברים, ספקי תוכן, מפרסמים ואפילו פעילים/פוליטיקאים.

לפי Tatar [1], קיימות שתי גישות עיקריות לניבוי פופולריות:

- טרום פרסום - גישה זו מנסה לחזות את הפופולריות של מאמר לפני פרסומו, תוך הסתמכות על מטא נתונים על תוכן המאמר, קישורים ועוד. למרות שגישה זו פחות מדויקת, היא שימושית יותר מכיוון שהיא מאפשרת שימוש במידע לביצוע שינויים במאמר לפני פרסומו. [2,3,4]

- לאחר פרסום - השיטה החלופית כוללת במודל החיזוי נתונים לגבי תשומת הלב שמקבל מאמר זמן קצר לאחר פרסומו. שיטה זו מדויקת יותר, וקלה יותר לניבוי. [5,6,7]

בנוסף, ניתן לחלק כל אחת משתי הגישות הללו לשתי גישות שונות לפי תוצאת החיזוי הצפויה:

- חיזוי מספרי - גישה זו שואפת לספק תחזית מספרית מדויקת של הפופולריות של המאמר. גישה זו השתמשה לאורך השנים בווריאציות שונות של הרגרסיה הליניארית (ניסוי להתאים לנתונים פונקציית חיזוי לינארית, המתבצעת בדרך כלל בעזרת חישוב

השגיאה בריבוע או השגיאה המוחלטת והרצת גרדיאנט למציאת הפרמטרים המספקים את מינימום השגיאות בסך הכללי) עם זאת, ניבוי מדויק של פופולריות נמצא פחות מדויק. [4,8]

- סיווג - על ידי התייחסות לבעיה כבעיית סיווג ניתן לפצל אותה ל-k קבוצות סיווג (לא חופפות) של טווחי פופולריות, במטרה לחזות נכון את טווח הפופולריות של הפריט בהנחה שטווח הפופולריות ידוע. בגישת הסיווג נעשה שימוש מספר מודלים נפוצים לאורך השנים [2, 3] : Random Forest (מסווג המשפר סיווג של עץ החלטה בודד על ידי שימוש במספר עצי החלטה החלטה שונים, אשר כל אחד מהם מבוסס על חלק מהתכונות, והסיווג בו יתבצע בסוף לפי החלטת רוב העצים); AdaBoost (מסווג המשתמש בקבוצה של מסווגים חלשים שכל אחד מהם מסווג תכונה נפרדת ובכל איטרציה בוחר מסווג אחד ומעניק לו משקל חדש, כאשר משקל המסווג גדל במקרה של שגיאה בזיהוי); KNN (סיווג לקבוצות על פי שכן קרוב, בו בכל פעם שמתקבלת דוגמא חדשה האלגוריתם ישייך אותה לקבוצה הנפוצה ביותר בקרב K השכנים הקרובים שלה) ; SVM ו Naive Bayes (עליהם יורחב בהמשך)

<u>מוטיבציה לבחירת המודלים</u>

מאמרי חדשות מקוונים פופולריים רק לפרק זמן קצר לפני שהם יורדים מסדר היום, ולכן אסטרטגיית "לאחר - פרסום" לא תניב תוצאות אפקטיביות במקרה זה. בנוסף, גישה זו פחות שימושית עבור כותבי תוכן באתרי חדשות ולכן בחרנו שלא להשתמש בה.

כפי שכתבנו, גישה מספרית פחות מדויקת מגישת סיווג, ולכן נעדיף לבחור בגישת סיווג. יתרה מכך, אין צורך בנתון מדויק במקרה זה, אלא די במידע אם המאמר עשוי להיות פופולרי כדי לספק את הערך הנדרש.

בחרנו במשימה הבינארית הקלאסית (סיווג מאמר לפי פופולרי/לא פופולרי) כדי לנסות לקבל תוצאות חיזוי מדויקות יותר באמצעות הכללה עבור שתי קבוצות בלבד.

מסד הנתונים שלנו שנוצר על ידי החוקרים Fernandes, Vinagre ו-Cortez. במאמרם הם בדקו חמישה מודלים לסיווג:
Random Forest; Adaptive Boosting; SVM with a Radial Basis Function kernel; K-Nearest Neighbors (KNN) and Naive Bayes

במסקנות שלהם הם מציינים ש-Random Forest משיג את התוצאות הטובות ביותר (נרשמו 67% דיוק). לאור זאת, בחרנו ליישם את SVM ו-Naive Bayes שלמדנו בקורס שלנו, שלפי החוקרים סיפקו אחוזי דיוק פחות טובים על מנת לחזות את דיוקם ביחס ל - Random Forest.

בנוסף, רצינו לעשות שימוש בעתיד בשני המודלים הנ"ל בפרויקטים אישיים, ולכן עובדה זו תרמה לבחירת המודלים הללו.

<u>חומרים ושיטות</u>

<u>סט הנתונים</u>

מקור: ttps://www.kaggle.com/yamqwe/predicting-number-of-shares-of-news-articles

מערך נתונים זה מסכם קבוצה הטרוגנית של תכונות על מאמרים שפורסמו על ידי Mashable בתקופה של שנתיים. המטרה היא לחזות את מספר השיתופים ברשתות החברתיות

(הפופולריות) של מאמר חדשותי. הנתונים נאספו במהלך תקופה של שנתיים, מה-7 בינואר 2013 ועד ה-7 בינואר 2015.

מספר מאמרים: 39797
מספר תכונות: 61

הערכים הינם ערכים מספריים/בוליאניים ולא מכילים טקסט

ערך החיזוי: ערך החיזוי הוא מספר רציף של שיתופים ברשתות החברתיות. אנחנו הפכנו אותו לבוליאני על ידי קביעת סף החלטה של 1400 שיתופים (ממוצע שיתופים למאמר על פני המאגר).

מודלים

השתמשנו בספריית Scikit learn לטובת ייבוא המודלים.

Gaussian Naive Bayes - GaussianNB - https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

SVM - https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

על כל מודל הרצנו בנוסף cross validation על 10 קבוצות על מנת לשפר את הדיוק של המודלים במודל ה -SVM בחנו שימוש בשלוש פונקציות kernel שונות על מנת לזהות את זו המספקת לנו את אחוז הדיוק הגבוה ביותר.

לא השתמשנו ב - Feature scaling על מנת לייעל את זמן הריצה של מודל הSVM משום שזמן הריצה היה זניח בשבילנו (לכל היותר מספר דקות).

לאור חוסר במשאבי זיכרון (RAM) נדרשנו להגביל את ריצת המודלים ל - 10,000 מאמרים בלבד.

SVM

מודל ה SVM היה מאוד פופולרי לפני פיתוח של רשת הנוירונים. בהינתן אוסף של נקודות השייכות ל - 2 סוגים שונים, SVM מוצא את המפריד הטוב ביותר ביניהם.
optimal separating hyperplane מוצא מישור שלם שמפריד בין הנקודות השונות
הבעיה ניתנת לפתרון על ידי שימוש במשוואות ובאמצעות שיטה שנקראת quadratic (QP programming)

Support Vectors

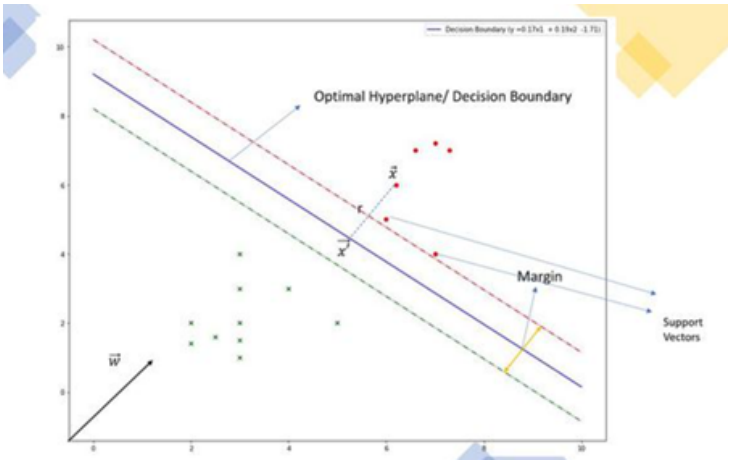
וקטורי תמיכה מהווים נקודות שקרובות לגבול ההפרדה, ולכן הכי קשה לסווג אותם.
בעזרתם SVM מספקת לנו מישור הפרדה אופטימלי על ידי מספר מצומצם של תכונות/פרמטרים.

Separating Hyperplanes

המפריד האופטימלי הוא מישור שלם החותך בין 2 קבוצות עם מאפיינים שונים.

במישור יכול להיות בכמה מימדים שונים והוא לא בהכרח לינארי.וקטורי התמיכה מהווים את ה"שוליים" של מישור ההפרדה ולכן כדי למצוא מפריד טוב ביותר נרצה לקחת את המישור שהוא במרחק מקסימלי מהשוליים.

SVM מגדיר מפריד טוב בהתאם למאפיינים של הנתונים



Non-Linear SVM

במציאות אנחנו מתמודדים עם נתונים מהווים לנו הפרדה לא לינארית, לכן השיטה של SVM לא לינארי היא נפוצה מאוד ועיקרה יצירת מפריד לינארי על ידי הגדלת המימדים.

הבעיה שנוצרת היא שהגדלת הממדים יוצרת חישובים ארוכים ומסובכים שמגדילים את סיבוכיות הזמן.

לכן אנחנו משתמשים במושג שנקרא "פונקציית kernel" שעושה סימולציה על החישובים. פונקציית ליבה לוקחת כקלט וקטורים במרחב המקורי ומחזירה את מכפלת הנקודות של הוקטורים במרחב.

ישנם מספר פונקציות קרנל נפוצות שניתן להשתמש בהם כדי לייעל את החישובים שלושת פונקציות הקרנל הנפוצות הן: Polynomial, RBF, Sigmoid.

- **Polynomial:** $K(x, x') = (x \cdot x' + c)^q$
- **RBF (radial basis function):** $K(x, x') = e^{-\frac{||x-x'||^2}{2\sigma^2}}$
- **Sigmoide:** $K(x, x') = tanh(\alpha x \cdot x' - b)$

<u>Naive Bayes</u>

סיווג נייב בייס הוא שיטת סיווג הסתברותית המתבססת על "חוק בייס" ועל ההנחה "הנאיבית" שאין תלות בין תכונות של אובייקט לאחר שהסיווג שלהם כבר ידוע.

המטרה שלנו היא לסווג אובייקט לאחת מ- k קטגוריות כאשר אשר האובייקט מאופיין על ידי וקטור התכונות $X = (x_1 ... x_n)$ ידועות ההסתברויות האפריוריות של הקטגוריות $p(C_K)$ לכל k, וגם ידועות ההסתברויות המותנות של כל תכונה בנפרד בהינתן הקטגוריה. הסתברויות אלה ניתנות להערכה מתוך מדגם מייצג. בנוסף מניחים את ההנחה ה"נאיבית" שאם ידועה הקטגוריה אזי התכונות אינן תלויות זו בזו.

כדי למצוא את הקטגוריה הסבירה ביותר משווים את ההסתברויות המותנות $p(C_k|x)$ ובוחרים את C_k שמוביל להסתברות המותנית המקסימלית. לפי חוק בייס:

$$p(C_k|x) = \frac{p(C_k)\, p(x|C_k)}{p(x)}$$

כיוון שהמכנה לא תלוי ב-k מספיק להשוות את המונים. בשל תכונה של ההסתברות המותנית, המונה שווה ל $p(C_K, x_1, ... x_n)$ לפי כלל השרשרת של הסתברות מותנית:

$$\begin{aligned}
p(C_k, x_1, \ldots, x_n) &= p(x_1, \ldots, x_n, C_k) \\
&= p(x_1 \mid x_2, \ldots, x_n, C_k) p(x_2, \ldots, x_n, C_k) \\
&= p(x_1 \mid x_2, \ldots, x_n, C_k) p(x_2 \mid x_3, \ldots, x_n, C_k) p(x_3, \ldots, x_n, C_k) \\
&= \ldots \\
&= p(x_1 \mid x_2, \ldots, x_n, C_k) p(x_2 \mid x_3, \ldots, x_n, C_k) \ldots p(x_{n-1} \mid x_n, C_k) p(x_n \mid C_k) p(C_k)
\end{aligned}$$

בשל ההנחה הנאיבית, הביטוי האחרון שווה ל-

$$p(x_1|C_k)\, p(x_2|C_k)\, \cdots\, p(x_n|C_k)\, p(C_k) = p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

ההסבר מציג את העקרונות הבסיסיים מאחורי מודל Naive Bayes.

עם זאת, קיימות וריאציות נוספות של המודל הנ"ל. אחת הפופולאריות שבהן ובה השתמשנו בפרויקט היא השיטה הגאוסיאנית (Gaussian naïve Bayes).

שיטה זו באה להתמודד עם העובדה שהמודל הבסיסי של Naive Bayes נועד לחשב הסתברויות של תכונות בדידות, ולכן אינו יכול להתמודד עם תכונות רציפות.

לאור זאת, במודל הגאוסייני מעריכים לכל תכונה רציפה את פונקציית ההתפלגות שלה, ובדרך כלל משתמשים לצורך כך בפונקציית ההתפלגות הנורמלית של גאוס, פונקציית צפיפות ההסתברות שלה (אשר מחליפה את ההסתברות המותנית במודל הבסיסי) מחושבת כך:

$$p(x = v \mid C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}}\, e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}}$$

על שני המודלים ביצענו cross validation, על מנת לנסות לשפר את דיוק הנתונים.

<u>סקירת המערכת</u>

קישור לפרוייקט: https://github.com/NeriyaZudi/PredictingNewsPopularityAI

<u>סקירת המערכת Predicting News Popularity using SVM</u>

המערכת פועלת באופן הבא:

1. בשלב הראשון מתבצעת קריאה של כל מאגר הנתונים מקובץ CSV
2. בשלב השני קביעת ערך בוליאני (פופולרי - 1 , לא פופולרי - 0) על ידי קביעת סף של כמות שיתופים מעל 1400 שיתופים - מאמר פופולרי, מתחת ל1400 שיתופים - מאמר לא פופולרי
3. לאחר מכן, מתבצעת הסרת מידע שלא רלוונטי למודל (לדוגמה: קישור URL של המאמר)
4. בשלב זה, המאגר מסודר לביצוע מודל SVM.
5. עקב גודל המאגר, נבצע אימון של המודל על חלק קטן מהמידע (מטעמי נוחות)
6. ניקח מספר של 5,000 מאמרים מכל קבוצה (פופולרי, לא פופולרי) וניצור מאגר חדש בעל 10,000 מופעים איתו נאמן את המודל שלנו.
7. בשלב זה, נבנה את המודל עם הפרמטרים הנדרשים X_train, X_test, Y_train, Y_test
8. נבצע אימון למודל (שימוש בAPI קיים)
9. לאחר שהרצנו את המודל, נציג את התוצאות ע"י מטריצת בלבול שמתארת את אחוזי ההצלחה של החיזוי שהמודל העניק לנו.
10. הצגת אחוז הדיוק שהמודל העניק לנו.
11. בשלב זה, יש לנו תוצאות שהמודל נתן לנו, אך אנו רוצים לבדוק האם יש אפשרות לשפר את התוצאות לכן נבצע תהליך של Cross-validation כדי להחליט באיזו פונקציית קרנל כדאי לנו להשתמש
12. הצגת תוצאות של Cross-validation
13. הצגת זמן הריצה של כל התוכנית
14. חשוב לשים לב! זמן הריצה משתנה בין הרצה להרצה.

<u>סקירת המערכת Predicting News Popularity using Naive Bayes</u>

המערכת פועלת באופן הבא:

1. בשלב הראשון מתבצעת קריאה של כל מאגר הנתונים מקובץ CSV
2. בשלב השני קביעת ערך בוליאני (פופולרי - 1 , לא פופולרי - 0) על ידי קביעת סף של כמות שיתופים מעל 1400 שיתופים - מאמר פופולרי, מתחת ל1400 שיתופים - מאמר לא פופולרי
3. לאחר מכן, מתבצעת הסרת מידע שלא רלוונטי למודל (לדוגמה: קישור URL של המאמר)
4. בשלב זה, המאגר מסודר לביצוע מודל Naive Bayes.
5. יצירת רשימה של כל העמודות (המאפיינים) הרלוונטיים כדי לקבל החלטה X , והעמודה Y שהיא מטרת החיזוי שלנו. (פופולרי, לא פופולרי)
6. בשלב זה, נבנה את המודל עם הפרמטרים הנדרשים X_train, X_test, Y_train, Y_test
7. נבצע אימון למודל (שימוש בAPI קיים)
8. נשמור את תוצאות החיזוי שהמודל העניק לנו.

9. לאחר שהרצנו את המודל, נציג את התוצאות ע"י מטריצת בלבול שמתארת את אחוזי ההצלחה של החיזוי שהמודל העניק לנו.
10. הצגת אחוז הדיוק שהמודל העניק לנו.
11. בשלב זה, יש לנו תוצאות שהמודל נתן לנו, אך אנו רוצים לבדוק האם יש אפשרות לשפר את התוצאות לכן נבצע תהליך של Cross-validation
12. הצגת תוצאות של Cross-validation (במקרה שלנו, ראינו שהמודל העניק אחוז דיוק טוב)
13. הצגת זמן הריצה של כל התוכנית
14. חשוב לשים לב! זמן הריצה משתנה בין הרצה להרצה.

דילמות/בעיות תכנותיות איתן נתקלנו:

בעיה טכנית : מחשב MacBook עם מערכת הפעלה MacOS Big Sur, לא תומך בהתקנת ספריות נחוצות כגון: pandas, sklearn
הפתרון: הרצת הקוד על מחשב אחר בסביבת עבודה PyCharm

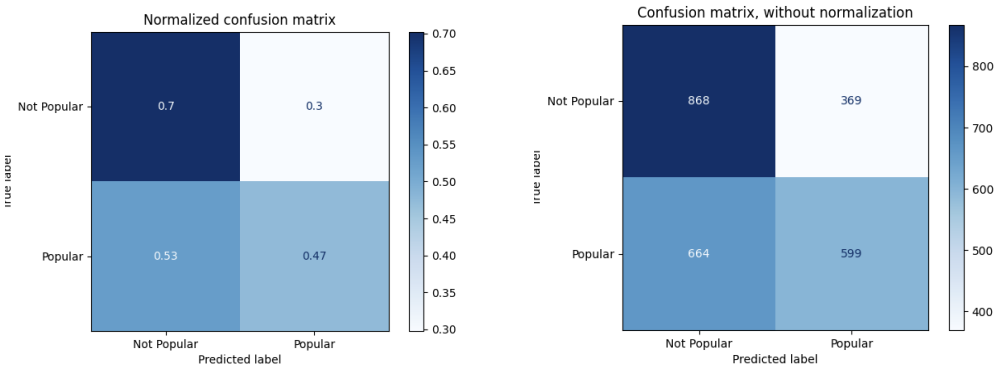
בעיה טכנית: עבודה עם מאגר נתונים בסדר גודל יחסית גדול, דרישות זיכרון גבוהות
הפתרון: אימון וביצוע המודל על חלק מהמאגר. כמו כן, ישנן פונקציות לציור גרפים שלא ניתן היה להשתמש בהן מפאת חוסר בזיכרון במחשב.

דילמה תכנותית: שימוש ב API קיים של השיטות שבחרנו, היה מאתגר להבין את השיטות לעומק מאופן השימוש

הפתרון: למרות השימוש ב API קיים, מימשנו את הקוד בצורה נוחה (תוך שמירה על גמישות ויעילות) ונתנו אפשרויות לשינוי ושליחת פרמטרים ע"י כתיבת פונקציות לכל שלב במערכת.

תוצאות

SVM



בחנו שלוש פונקציות kernel שונות:

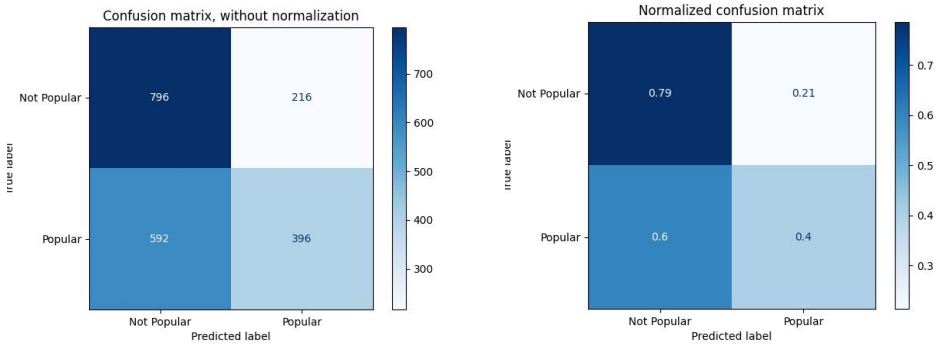
```
Kernel : rbf
SVM accuracy after 10 fold CV: 0.58 (+/- 0.04), 87.5s
```

```
Kernel : poly
SVM accuracy after 10 fold CV: 0.58 (+/- 0.04), 42.5s
```

```
Kernel : sigmoid
SVM accuracy after 10 fold CV: 0.51 (+/- 0.03), 32.477s
```

פונקציית ה -RBF הניבה את התוצאה המדויקת ביותר ולכן בחרנו בה לטובת התוצאות.

Naive Bayes




```
The percentage of accuracy is 59.599999999999994  %
```

השוואות ומסקנות

נשווה כעת את ביצועי שני המודלים לאור מספר פרמטרים מרכזיים:

דיוק

הפרמטר הראשון והפשוט ביותר להשוואה בין המודלים הוא אחוז הדיוק המתקבל מכל מודל.

באופן כללי מהמחקר עולה כי מודל Naive Bayes נוטה להשיג אחוזי דיוק נמוכים יותר ממודלים אחרים. עם זאת, בפרוייקט שלנו פרמטר הנ"ל מודל Naive Bayes השיג אחוז דיוק מעט טוב יותר (0.595) ממודל ה SVM ´(0.58).

סיבוכיות זמן

בפרויקט הנוכחי סיבוכיות הזמן של שני המודלים הייתה יחסית זניחה (מספר דקות לכל היותר), אבל בהינתן מאגר מידע גדול יותר, סיבוכיות הזמן של מודל SVM תתחיל לעלות בצורה אקספוננציאלית לעומת סיבוכיות הזמן של מודל Naive Bayes שהיא לינארית, משום שהמודל עצמו מבצע כפל פשוט בין המשתנים לאחר שידועה טבלת ההסתברויות.

במקרים כאלו היה נכון יותר להשתמש ב"התאמת משתנים" (Feature scaling), על מנת לשאוף לצמצם את זמן הריצה (דבר שלא נעשה בקוד שלנו)

סיבוכיות מקום

סיבוכיות המקום של מודל Naive Bayes נמוכה בצורה משמעותית ביחס ל SVM. במודל Naive Bayes אנו לא נדרשים לשמור את הנתונים בעת הרצת המודל, אלא רק לחשב את ההסתברויות

המותנות שלהם, ולשמור טבלה של ההסתברויות המותנות, בעוד מודל ה - SVM דורש שמירה של הנתונים ב - RAM מה שהוביל בפרויקט שלנו למגבלה על כמות הנתונים עליהם יכולנו להריץ את מודל ה - SVM ולהגבלת המאגר ל - 10000 מאמרים בלבד.

<u>יכולת הרחבה (scalability)</u>

מודל SVM אינו ניתן להרחבה בקלות. כל הוספה של תכונה חדשה למארג התכונות תדרוש הרצה מחדש של כל המודל על מנת לקבל מפריד חדש. לעומת זאת במודל הנייב - בייס הוספת תכונות נוספות לא תדרוש הרצה מחודשת של כל המודל, אלא רק הוספה של ההסתברויות המותנות אל טבלת ההסתברויות הנתונה.

<u>עלות מתכנת</u>

בפרמטר עלות המתכנת ישנם מספר אלמנטים משפיעים המטים את הכף לטובת מודל Naive Bayes:

ראשית, כפי שהוזכר לעיל, במקרים של זמן ריצה גבוה במודל ה SVM נוצר צורך להשתמש בFeature scaling דבר שדורש ביצוע פעולות נוספות בקוד ומוסיף ל"עלות המתכנת".

שנית במודל ה SVM על מנת לקבל מפריד לינארי טוב יותר, נהוג להשתמש בפונקציית kernel (כמו שנעשה בפרוייקט שלנו), יש לנסות לבחור את פונקציית kernel הטובה ביותר, דבר שמוסיף לעלות המתכנת.

שלישית, מימוש מחדש של מודל Naive Bayes הוא פשוט יחסית ומבוצע בזמן קצר. לעומת זאת מימוש מחדש של מודל ה SVM הוא מורכב יותר ודורש זמן רב יותר.

<u>זיהוי תלויות המשפיעות על הנתונים</u>

אמנם לא ראינו את הפרמטר בא לידי ביטוי בתוצאות, אבל באופן עקרוני מודל Naive Bayes כהנחת יסוד מניח אי - תלות בין התכונות השונות. לעומת זאת מודל ה - SVM כן מתייחס לתלויות השונות בין התכונות. במאגר הנתונים שלנו קיימות תלויות בין התכונות השונות אשר Naive Bayes יתעלם מהן, כדוגמת התלות בין נושא המאמר לערוץ בו פורסם, או היחס בין ביטויים חיוביים ושליליים בתוך המאמר, ואשר SVM עשוי לזהות ולתת להן ביטוי בסיווג.

<u>בעקבות הפרויקט עלו לנו מספר תובנות חשובות:</u>

<u>הבנת תחום הבינה המלאכותית</u>

ראשית, נראה כי למרות מה שניתן לחשוב בצורה ראשונית כאשר ניגשים לתחום הבינה המלאכותית, ישנם נושאים אשר גם בהינתן כמות משמעותית של נתונים ושימוש במודלים שונים ומתקדמים, לא ניתן לחזות אותם בצורה טובה מספיק, כפי הנראה מתוצאות הפרויקט שלנו ומתוצאות מחקר נוסף [2] אשר לא הראו אחוזי דיוק גבוהים משמעותית (0.67).

בנוסף ניתן לראות מהפרוייקט הנ"ל כי שימוש בשיטות שונות לא מוביל לשינויים של עשרות אחוזים ברמת הדיוק. אי אפשר להשליך את הנתון הזה על בעיות אחרות, אולם ניתן להניח לבחינה עתידית את ההשערה (אשר תידרש אישוש ממחקרים אחרים) כי דרך יעילה יותר להבנה האם ניתן לחזות נתון מסויים ממאגר נתונים תהיה להריץ עליו מודלים פשוטים למימוש כדוגמת Naive Bayes במקרה הנ"ל כ"קריאת כיוון" לפני שמתחילים לממש מודלים מורכבים יותר אשר עשויים לגלות רק לאחר עבודה מאומצת כי לא ניתן לחזות את הנתון המבוקש ממאגר הנתונים.

למדנו כי קיים קשר הדוק בין מאפייני הנתונים לבין המודלים אותם אנו מבקשים להחיל עליהם, ולאור זאת יש להבין היטב את התכונות הקיימות במאגר הנתונים, ולשקול את ההשפעה שלהם על דיוק המודלים (לדוג׳ ההשפעה של שימוש בתכונות רבות אשר מקיימות תלות במודל כמו Naive Bayes)

לאור זאת, התחדדה אצלנו ההבנה של חשיבותה האקוטית של עבודת המהנדס/מדען הנתונים להבנה מעמיקה בקשר שבין המודלים ובין המידע הנתון, אשר היא מכרעת על הצלחתו בפתרון הבעיה (ואולי ניתן לומר שהיא אף הפרמטר המרכזי בה).

חשיבות סיבוכיות המקום

למרות החזרות הרבות בשיעורי הקורס על חשיבותה האקוטית של סיבוכיות מקום בתחומי הבינה המלאכותית, השיעור הפרקטי אשר למדנו מבעיית סיבוכיות המקום אשר לא אפשרה לנו להריץ את אלגוריתם SVM על מאגר המידע המלא, אלא רק על רבע ממנו העמיק בנו חשיבות זו בצורה משמעותית. למדנו כי למרות האפשרות למודלים מתקדמים לספק אחוזי דיוק גבוהים, ההתחשבות בסיבוכיות המקום היא פרמטר משמעותי ולפעמים אף מכריע בשימוש במודלים מדויקים פחות אך ניתנים ליישום.

שימוש בגישות שונות

הבנו את חשיבות השימוש בגישות שונות לפתרון בעיה, וההשפעה שלהן על התוצאה הסופית ובעיקר נחשפנו לכוח העצום של סקירת הספרות האקדמית והכרת הגישות השונות לפתרון הבעיה שכבר היו בשימוש והתובנות העולות מהן, כך שכאשר אנו מגיעים לפתור בעיה, לרוב איננו מתחילים מאפס, אלא יכולים וצריכים לבנות נדבך נוסף על המידע הקיים.

מקורות:

1. Alexandru Tatar, Marcelo Dias de Amorim, Serge Fdida, and Panayotis Antoniadis. A survey on predicting the popularity of web content. Journal of Internet Services and Applications, 5(1):1-20, 2014.

2. K. Fernandes, P. Vinagre and P. Cortez. A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News. Proceedings of the 17th EPIA 2015 - Portuguese Conference on Artificial Intelligence, September, Coimbra, Portugal

3. Tsagkias M, Weerkamp W, De Rijke M (2009) Predicting the volume of comments on online news stories. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management. ACM, Hong Kong, China, pp 1765-1768

4. Bandari R, Asur S, Huberman BA (2012) The pulse of news in social media: Forecasting popularity. In: ICWSM. The AAAI Press, Dublin, Ireland

5. Mohamed Ahmed, Stella Spagna, Felipe Huici, and Saverio Niccolini. A peek into the future: predicting the evolution of popularity in user generated content. In Proceedings of the sixth ACM international conference on Web search and data mining, pages 607-616. ACM, 2013.

6. Andreas Kaltenbrunner, Vicenc Gomez, and Vicente Lopez. Description and prediction of slashdot activity. In Web Conference, 2007. LA-WEB 2007. Latin American, pages 57-66. IEEE, 2007.

7. Jong Gun Lee, Sue Moon, and Kav´e Salamatian. Modeling and predicting the popularity of online contents with cox proportional hazard regression model. Neurocomputing, 76(1):134-145, 2012.

8. Szabo G, Huberman BA (2010) Predicting the popularity of online content. Commun ACM 53(8):80-88