למידה עמוקה עבודה 3

Data exploration:

הבעיה: הנתונים מתייחסים לנתוני חיפוש ותוצאות שלקוחות ביצעו ברשת Home Depot. עבור כל רשומה בנתונים הכוללת את הכותרת של המוצר אותו חיפש הלקוח ותיאור התוצאה שהתקבלה דירגו את הרלוונטיות של התוצאה לחיפוש ע"י ביצוע מבחן בו 3 שופטים התבקשו לדרג את הרלוונטיות של התוצאה לחיפוש ולבסוף לקחו את הממוצע של שלושת הדירוגים. כך קיבלנו את ערך הרלוונטיות לכל רשומה שנע בטווח 1-3 כאשר יש בדיוק 13 ערכים אפשריים. ניתן להתייחס לבעיה כבעיית רגרסיה בכך שיש לחזות את הרלוונטיות של התוצאה לשאלה על ידי מספר בין 1 ל-3, לעומת זאת ניתן להתייחס לבעיה כבעיית סיווג בכך שיש לסווג כל זוג של חיפוש-תוצאה לאחד מ-13 ערכי הרלוונטיות השונים. אנו ניסנו להתייחס לבעיה ב-2 האופנים האלו כפי שנתאר בהמשך.

הנתונים מחולקים ל-2 קבצים עיקריים- train, test כך שtrain מכיל 74067 רשומות וtest מכיל 166693. בהסבר על הנתונים בkaggle היה נתון שיש חיפושים בtest שכבר המודל ראה בעבר בtrain אך גם כאלו שלא ראה בעבר. בדקנו מהו מספר הsearch\_items הייחודיים בtrain ובtest:



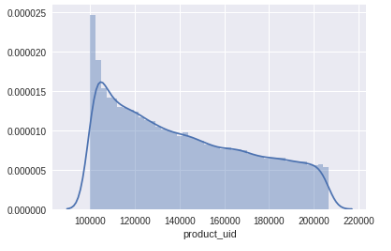
זאת אומרת שבוודאות נתקל בלפחות 10,632 ביטויי חיפוש ב-test שלא נתקלנו בהם בעבר ונצטרך לחזות בצורה הטובה ביותר את ההתאמה שלהם לתוצאת החיפוש שלהם.

ה-label של כל זוג (ביטויי חיפוש, תוצאות חיפוש) הוא בעצם מידת ההתאמה של תוצאת החיפוש לביטוי החיפוש המבוטאת במספר בין 1 ל-3 כאשר 3 מתאר את מידת ההתאמה הגבוהה ביותר ו-1 מתאר את מידת ההתאמה הנמוכה ביותר. כלומר מדובר במשימת **חיזוי**.

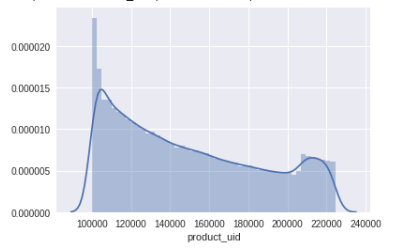
לא קיימים ערכים חסרים בנתונים.

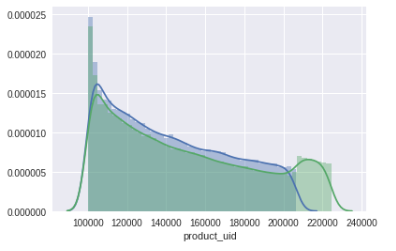
התפלגות הנתונים:

* התפלגות המוצרים שהחיפוש מחזיר בקובץ train:



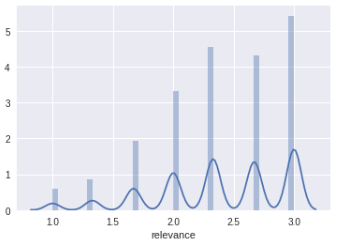
* התפלגות המוצרים שהחיפוש מחזיר בקובץ test:



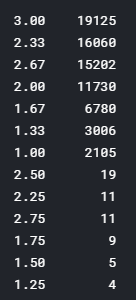


ניתן לראות שהתפלגות המוצרים שהחיפוש מחזיר בקבצי הtrain וה-test דומה כלומר כנראה שאנשים מחפשים לרוב את אותם המוצרים או לחלופין שהמערכת מחזירה לרוב את אותם המוצרים גם עבור חיפושים שונים (פחות סביר מאחר וציון ההתאמה של רוב החיפושים הוא לרוב ציון טוב) או שילוב של השניים.

בדקנו כיצד מתפלגים ציוני ההתאמה של המוצרים:



כפי שציינו לעיל, רוב החיפושים מסתיימים בתוצאות מאוד טובות (ציון התאמה בטווח 2.5-3), חלק גדול מהחיפושים מסתיימים בתוצאה בינונית (ציון התאמה בטווח 2-2.5) ומעט חיפושים מסתיימים בתוצאות לא טובות (ציון התאמה בטווח 1-2). זאת אומרת ש**הנתונים הם imbalanced** ולכן בהמשך נשתמש בשיטות להתמודד עם בעיה זו. כמות סיווגים לכל class:

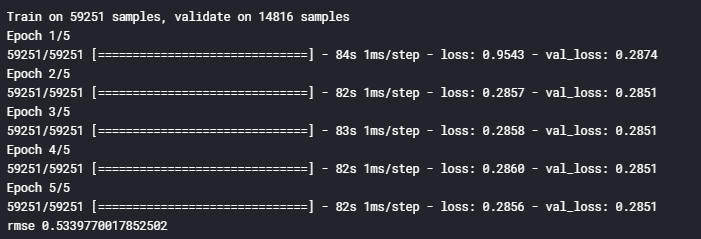


ניתן לראות שמעט מאוד רשומות קיבלו את הדירוגים: 2.5, 2.25, 2.75, 1.75, 1.75, 1.5, 1.25

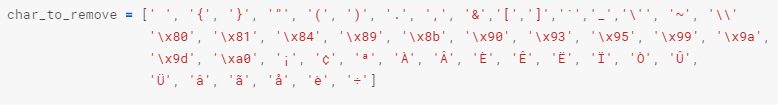
**חלק 1- Character level LSTM**

Preprocess:

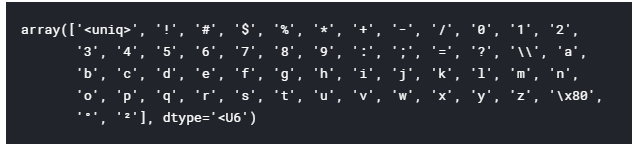
הערה: בראשית הדרך חיברנו בין שדה title ושדה product\_description עבור כל מוצר על מנת לנסות לקבל כמה שיותר מידע על המוצר. עם זאת לאחר התייעצות עם נתי ומאחר והריצה הייתה מאוד ארוכה עד כדי כך שלא הצלחנו לקבל תוצאות בזמן סביר (בהתייחסות לכל התווים ללא חיתוך לפי ממוצע וכו') החלטנו לא להשתמש בשדה product\_description על מנת שנוכל לשפר את המודל ולנסות מודלים נוספים הרצים בזמן סביר מאשר הרצה של מודל עיקרי אחד בזמן לא סביר. בנוסף ניסינו להוסיף חלק מה- product\_description(לקחנו את כל 400 התווים הראשונים) אך זה לא שיפר את תוצאות המודל וגרם לכך שזמן האימון יהיה ארוך יותר. להלן דוגמה לאחד ה-epoches שהתקבלו:



ראשית יצרנו data frame בשם all\_df שהיא בעצם חיבור של שתי ה-data frames של הtrain והtest על מנת להכין את הנתונים על בסיס תווים הנמצאים בכל הנתונים. חילקנו את הטקסט בשדות search\_trem ו-proudct\_title לתווים והסרנו תווים שלאחר מעבר על הנתונים הגענו למסקנה שאינם מוסיפים מידע אלא עשויים לגרום לרעש. התווים אותם הסרנו:



התווים הייחודים שנותרו:



את התו “<uniq>’ אנו הוספנו על מנת לתאר את ה-padding ב-0 שיתווסף בהמשך.

לאחר החלוקה לתווים המרנו כל תו למספר באמצעות label encoder ובכך קיבלנו עבור כל רשומה שתי מערכים של מספרים, אחד עבור הsearch\_term ואחד עבור הproduct\_title.

כעת בדקנו מהו האורך המקסימלי של כל המערכים שהתקבלו (מספר תווים מקסימלי) תחת הפיצ'רים search\_term ו-product\_title ובכך קיבלנו שתי ערכים prod\_max\_len=128 ו-search\_max\_len=58.

ביצענו padding של אפסים בתחילת כל מערך על מנת שכל המערכים השייכים לאותו פיצ'ר יהיו באותו האורך. קבענו שאורך המערכים יהיה כאורך המקסימלי (כלומר prod\_max\_len ו- search\_max\_len בהתאמה). יש לציין שגם ניסינו לבצע padding לאורך הממוצע של המערכים (וחיתוך למערכים הארוכים מהאורך הממוצע) אך קיבלנו תוצאות פחות טובות ולכן בחרנו להשאיר את ה-padding לאורך המקסימלי.

כעת בנינו את המודל הראשון והבסיסי:

שיטת ולידציה: חלוקה של קובץ הtrain הנתון לtrain ו-validation כאשר גודל הvalidation הוא 0.2 מגודל הtrain.

input product\_title

input search\_term

LSTM

LSTM

concatenate

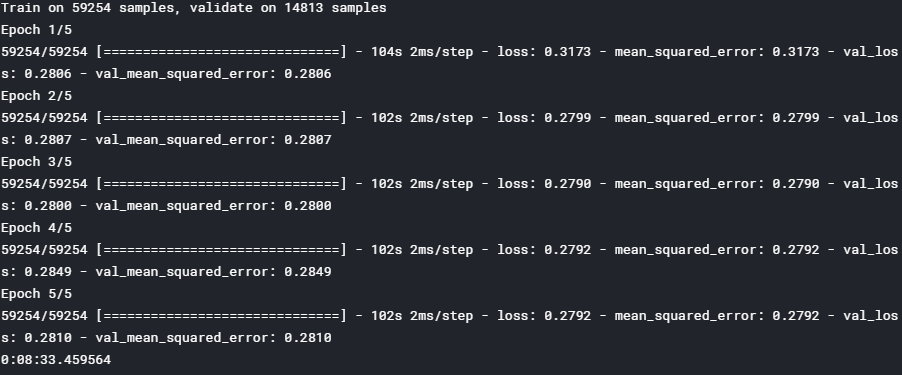
Dense

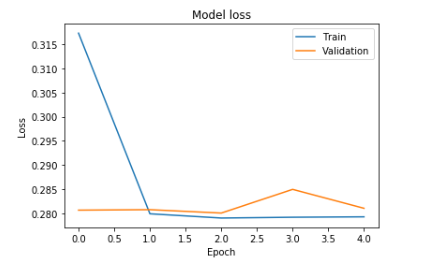
output

(value 1-3)

יש לציין שבמודל יש שכבת LSTM אחת אך היא מקבלת 2 inputs שונים ולכן גם יש לה 2 outputs שונים ולכן המודל משורטט כאילו מדובר ב-2 מודלים שונים שמתמזגים למודל אחד (בשלב ה-concatenate). אנו מציינות זאת מאחר וזה לא היה לנו ברור מאילו מהמאמר ולכן אנו חושבות שחשוב לציין זאת.

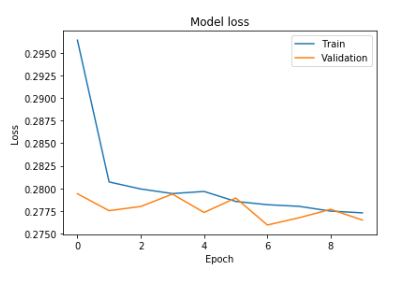
לאחר אימון הרשת על עם epoch=5 ו-batch\_size=72 קיבלנו:





ניתן לראות שיש שיפור בערך ה-loss של הtrain בתהליך הלמידה אך מה שחשוב הוא ה-loss של ה-validation והוא לא משתפר, ומתחיל מערך יחסית נמוך. ננסה לשפר את המודל.

ניסינו להגדיל את מספר הנוירונים בשכבת ה-LSTM מ-50 ל-100 והגדלנו את מספר ה-epoches מ-5 ל-10 אך עדיין אין ירידה משמעותית באופן הדרגתי עבור הloss של הvalidation:



כעת הסתכלנו על הבעיה ב-2 דרכים שונות עם שיטות ולידציה שונות:

1. בעיית רגרסיה (כפי שהצגנו לעיל) עם k-fold cross validation.

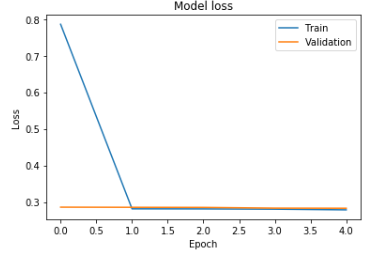
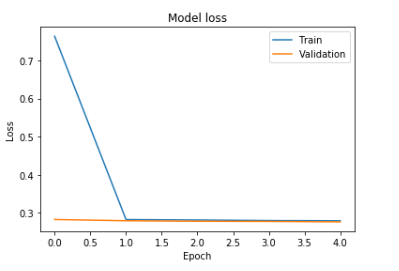
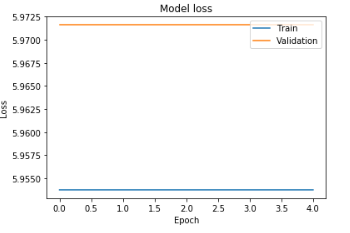
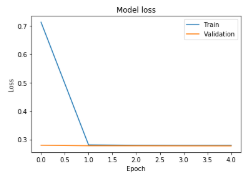
2. בעיית סיווג עם StratifiedKFold - ניתן לעשות זאת מכיוון שמספר הערכים הניתנים במשתנה המטרה relevance שווה ל-13 ערכים שונים אפשריים. הסיבה לכך היא מאופן חישוב הrelevance- נבחרו 3 שופטים לדירוג הרלוונטיות של המוצר לחיפוש שנעשה וה-relevance נקבע לפי ממוצע הדירוגים של 3 השופטים. במקרה זה נשתמש ב-StratifiedKFold כדי לאמוד את ביצועי המודל שלנו כיוון ששיטה זו מבצעת פיצול של הנתונים ל-folds כך שהתפלגות הנתונים ב-folds השונים תהיה זהה להתפלגות הנתונים ב-data המקורי ואכן זה מתאים לבעיה שלנו מאחר ויש imbalance בנתונים כפי שהצגנו בתחילת הדוח.

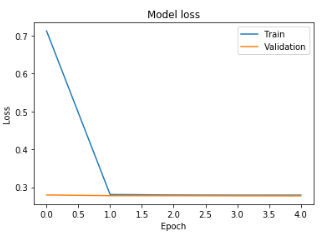
3. בעיית רגרסיה עם StratifiedKFold - הסיבה שלא השתמשנו ב- StratifiedKFold עבור בעיית הרגרסיה ב-1 היא ששיטת ולידציה זו דורשת שה-class לא יהיה מספר רציף. החלטנו לנסות להשתמש בשיטה זו גם עבור בעיית רגרסיה ע"י המרת y למספרים באמצעות label\_encoder רק עבור הפיצול ע"י StratifiedKFold ושימוש בy המקורי עבור אימון המודל.

תוצאות שיטה 1:

מודל כפי שהוסבר ושורטט לעיל רק שבשכבה האחרונה לא הגדרנו פונקציית אקטיבציה ופונקציית ההפסד היא mse.

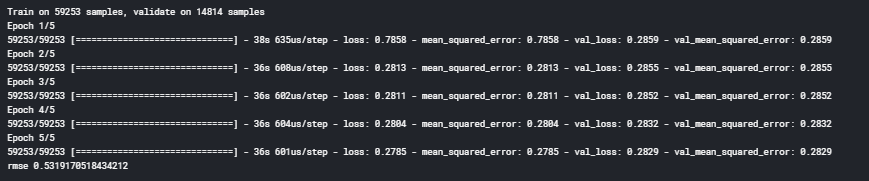
להלן גרפים המתארים את הloss בכל אחד מחמשת ה-epochs שביצענו:



ניתן לראות שבfold השלישי הloss גרוע מאוד. ההנחה שלנו היא שמאחר והנתונים הם imbalanced אז ספציפית הרשומות שנבחרו עבור הfold הזה אינן טובות בכך שלוקח למודל יותר זמן לעבד אותן ואולי לא ניתן למצוא דפוסים ברשומות אלו בכלל (מאחר וה-loss של הtrain גם לא טוב).

יש לציין כי בfolds האחרים יש מעט שיפור בloss של הvalidation אך לא רואים זאת בבירור בגרף מאחר ובהתחלה הloss על הtrain היה משמעותית יותר גבוה, לדוגמה פלט של אחד מהfolds שהרצנו:

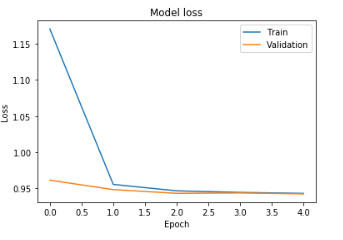
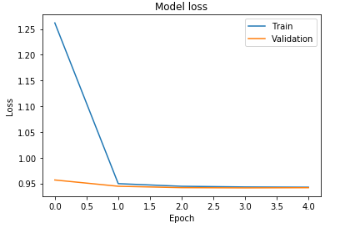


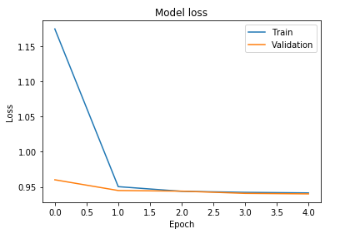
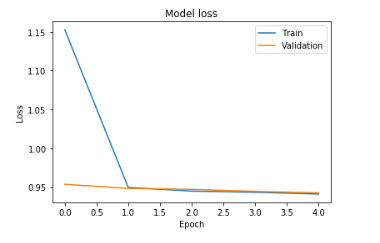
ניתן לראות שקיים שיפור בloss של הvalidation

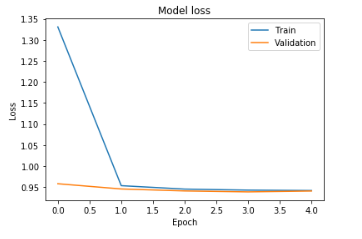
|  |  |
| --- | --- |
| rmse | fold |
| 0.531 | 1 |
| 0.525 | 2 |
| 2.443 | 3 |
| 0.526 | 4 |
| 0.525 | 5 |
| 0.91 | mean |

תוצאות שיטה 2:

מודל כפי שהוסבר ושורטט לעיל רק שבשכבה האחרונה פונקציית האקטיבציה היא softmax, פונקציית ההפסד היא categorical\_crossentropy.

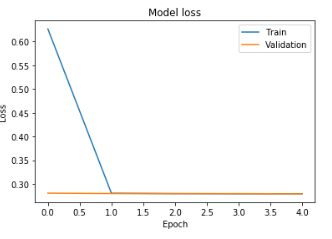
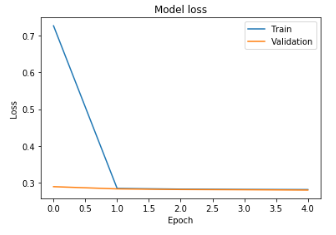


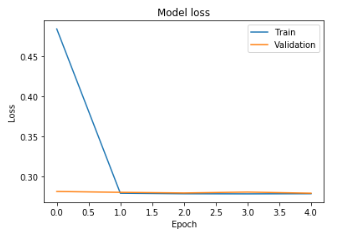
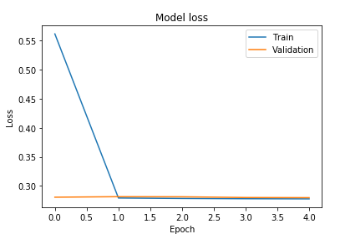
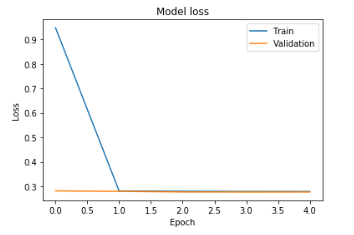


|  |  |
| --- | --- |
| accuracy | fold |
| 0.581 | 1 |
| 0.581 | 2 |
| 0.581 | 3 |
| 0.580 | 4 |
| 0.581 | 5 |
| 0.581 | mean |

תוצאות שיטה 3:

מודל כפי שהוסבר ושורטט לעיל רק שבשכבה האחרונה לא הגדרנו פונקציית אקטיבציה ופונקציית ההפסד היא mse.



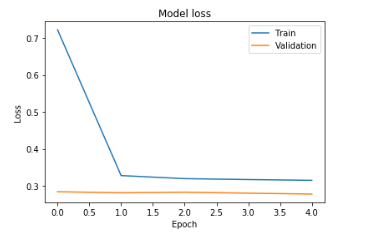
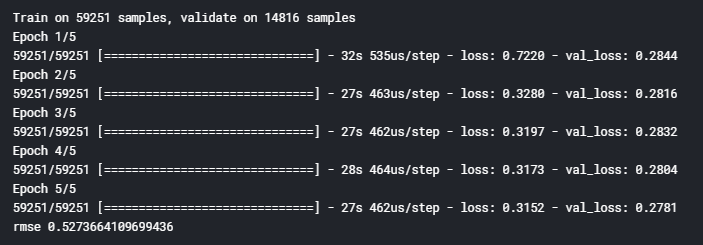
|  |  |
| --- | --- |
| rmse | fold |
| 0.528 | 1 |
| 0.529 | 2 |
| 0.528 | 3 |
| 0.528 | 4 |
| 0.526 | 5 |
| 0.528 | mean |

ניתן לראות שאכן בשימוש ב- StratifiedKFold עבור בעיית רגרסיה אנו כבר לא מקבלות את המקרה בו ה-loss היה קבוע וגבוה מאוד עבור הtrain והvalidation כפי שראינו במקרה 1 שבו השתמשנו בk-fold וחשדנו שמצב זה התקבל מהחלוקה הלא הוגנת מבחינת ערך המטרה בשימוש בk-fold.

**שיפורים-** נמשיך עם הגישה השלישית כלומר התייחסות לבעיה כבעיית רגרסיה ושימוש בשיטת ולידציה StratifiedKFold:

ניסנו לשפר את המודל ע"י הוספת שכבת dense עם 50 נוירונים לאחר שכבת הconcat שלאחריו מגיע גם שכבת dropout (עם הסתברות 0.3) על מנת למנוע over fitting ואחרי שכבת הdropout שכבת dense אחרונה שהיא הoutput.

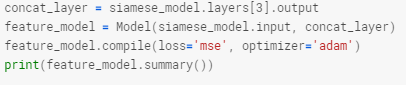
קיבלנו שה-rmse לא השתפר. בנוסף הloss של הtrain פחות טוב מה-loss של הvalidation לאורך כל תהליך האימון כתוצאה מהוספת שכבת ה-dropout. דוגמה לepoche אחד בתהליך ה-cross validation:

Feature extraction

מאחר והשתמשנו בcross validation, אימנו את המודל כל פעם על חלק אחר של הנתונים ולכן כעת אין לנו מודל אחד שהתאמן על כל הנתונים. לכן ראשית ניצור מודל כזה (הזהה למודל שהרצנו בכל fold רק שכעת נריץ אותו על כל נתוני הtrain).

לאחר אימון המודל, שלפנו את השכבה השלישית (שזוהי שכבת הconcat) ונתייחס אליה כפלט של המודל הנועד לקבלת הfeatures. את הקלט של המודל נשאיר כמו שהוא כלומר:



כעת נחזה את ערך ה-train וערך ה-val ונבדוק את הrmse שהתקבל עבור החיזוי ע"י כל אחד מהמודלים: XGBRegressor ו- LGBMRegressor כאשר הקלט לאותם מודלים הוא הפלט שהתקבל מהמודל feature\_model שתיארנו לעיל וערכי הlable כלומר הy\_data.

לאחר ביצוע tuning לכל אחד מהמודלים על-פי ערך הrmse שהתקבל עבור ה-validation set, הרצנו את המודלים האלו על כל נתוני הtrain set והtest set הנתונים, בתהליך כפי שתואר לעיל (קודם חילוץ הפיצ'רים ממודל הSiamese ולאחר מכן הכנסתם למודל הml) והגשנו submission של מודלים אלו.

**חלק 2- word embeddings and word level LSTM**