

**Data Collection, Sensing and Dataset Creation**

יהונתן להט 308147248

אלון קרמונה 312469083

לימוד מכונה פרויקט חלק א'

‏2020

תוכן עניינים

Data collection and Sensing ...................................................................................................2

Dataset Creation

Exploratory data analysis .1 ...........................................................................................3-7

Data Creation.2

Pre processing .........................................................................................................8-9

Segmentation ............................................................................................................. 9

Feature Extraction ................................................................................................. 9-11

Feature representation ............................................................................................. 11

Feature Selection...................................................................................................... 11

Dimensionality Reduction ........................................................................................ 11

…………………….…………………………………………………………….. Model Training 11

נספחים ..............................................................................................................................12-16

Data collection and Sensing

בתהליך בניית מערכת לימוד המכונה קיימים מספר שלבים, השלב הראשון הוא שלב ה-Data Collection, אוסף הנתונים.  
בשלב זה נבחרים הישויות/ יחידות/ דוגמאות עליהן נרצה לאסוף את המידע שעל פיו יתבסס המודל.  
אנחנו נרצה שהנתונים אותם נאסוף יהיה נתונים ממצים, אמינים ואם ניתן מתויגים (labeled), שלב זה הינו הבסיס לתהליך לימוד מכונה מוצלח.

כחלק משלב ה-Sensing, שלב החישה, נאספו הנתונים המתאימים לישויות משחקי המחשב. סוג החישה הנעשה הינו סטטי שכן הוא קבוע ונטול מאפייני זמן. הנתונים נדגמו פעם אחת עבור כל ישות – שכן מדובר בערכים שאינם משתנים ביחס הזמן.

כמובן שניתן היה להוסיף מדידות הן סטטיות והן דינמיות שהיו עשויות לסייע למודל.  
דוגמה למדדים סטטיים הוא נתון אשר מתייחס לרזולוציית התצוגה של משחקי המחשב או למחיר בו נמכר, מהיכרותנו את התחום מדדים אלו עשויים להשפיע על התנהגות הצרכן.

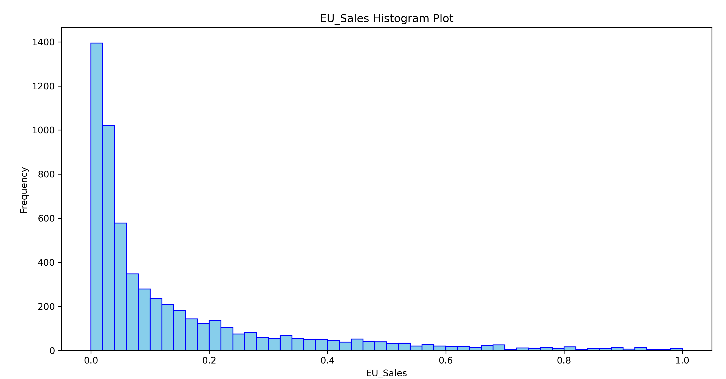
מדד דינאמי אותו היה ניתן להוסיף הוא השוואה של המכירות באזורים השונים או כמות חיפושי האינטרנט של המוצר – בחודש הראשון להוצאתו, בשנה הראשונה ובשנה העוקבת, מדד זה ניתן להציג באמצעות וקטור.

הנתונים אותם קיבלנו ושעליהם נבסס המודל הינם אודות משחקי מחשב שונים, הישויות מתויגות שכן ידוע לנו כמות המכירות שלהן באירופה. קטגוריית משימת הלמידה הינה Supervised Learning -למידה מונחת, בקטגוריה זו תיוג הדאטה ידוע והשימוש בו הוא למשימות סיווג או חיזוי. במשימה זו אנו ננסה לנסות לחזות את כמות המכירות של משחק באירופה על בסיס נתונים אחרים שנאספו.  
מתוך נתונים שחלקם נומריים וחלקם קטגוריאליים נרצה לתת הערכה מספרית לכמות המכירות ועל כן, מדובר במשימת חיזוי ונזדקק לערכים של ה-Samples ב-Training Set בכדי לקבל אותו.

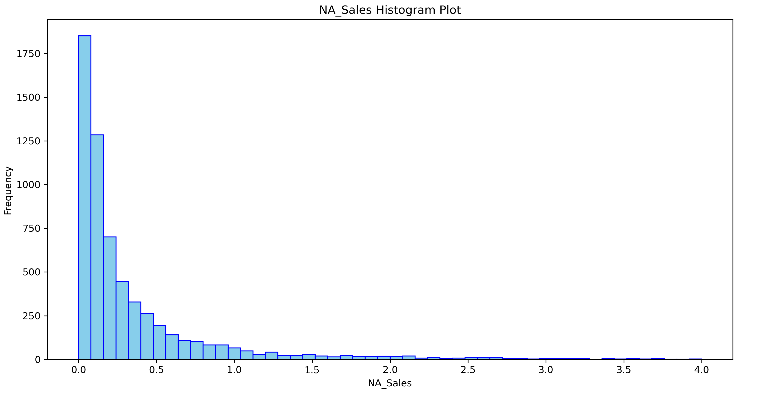
באמצעות הנתונים אותם קיבלנו היה ניתן לבצע סוגים נוספים של משימות למידה. לדוגמה משימת סיווג, יכולנו לבנות מודל שמטרתו היא סיווג קטגוריית המשחק על פי מאפייני המשחק, דירוגו ונתוני מכירותיו. מודל זה יכל לסייע לחברות המפתחות את המשחקים, כחלק מהתהליך העסקי.

Dataset Creation

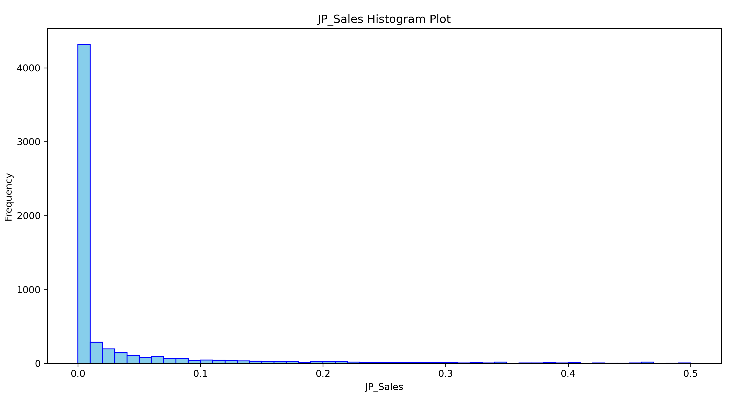
Exploratory data analysis **- משתנים אפריוריים:**

* משתנה מטרה- EU sales

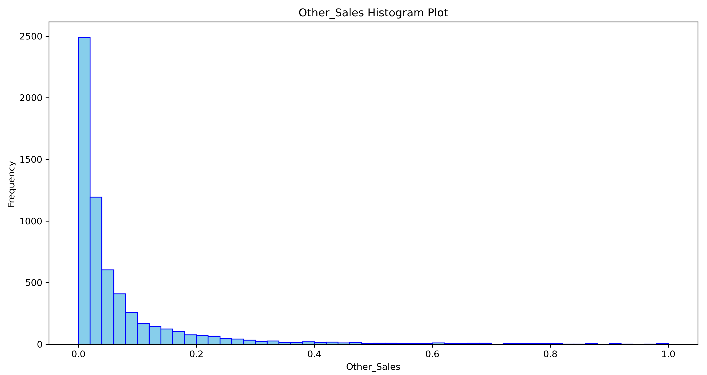
זוהי למעשה ההיסטוגרמה של המשתנה המוסבר אליו אנו ננסה לשאוף באמצעות המודל, אותו נאמן בעזרת הפיצ'רים המסבירים האחרים. משתנה זה מייצג את כמות המכירות עבור כל אחד ממשחקי המחשב באירופה. הטווח נע בין 0-28.96 כאשר אנו יכולים לראות מן ההיסטוגרמה כי ההתפלגות אינה אחידה אלא יורדת בצורה אקספוננציאלית ככל שעולים בערכי כמות המכירות (ציר הX), וכן שרוב הDATA שלנו מתרכז בצד השמאלי של הגרף בצמוד לערכים הנמוכים. לכן גרף זה אינו מאוזן, אך דבר זה מתיישב עם המציאות.

* NA Sales

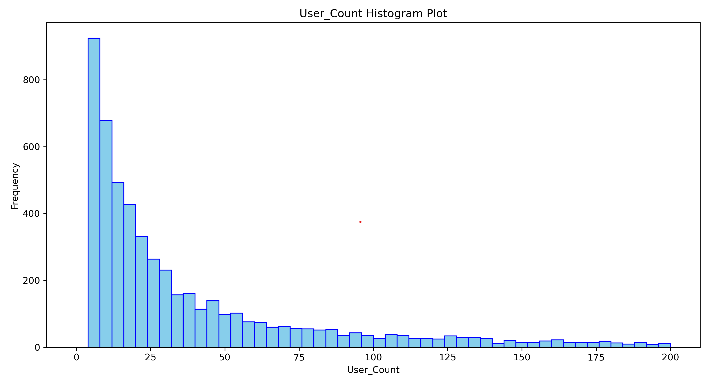
משתנה רציף זה מתייחס לכמות המכירות של כל משחק בצפון אמריקה. בדומה למשתנה המוסבר, המשתנה מייצג מכירות על סמך מיקום גיאוגרפי שבעזרתו אנו עשויים לתת אינדיקציה טובה בהמשך למשתנה המוסבר בזכות התנהגות משתמשים דומה במדינות/יבשות בעל אופי דומה. טווח הערכים נע בין 0-41.36 , אך הרוב המוחלט מתכנס לערכים הנמוכים בין 0 ל1.5, כאשר מגמת ההתפלגות היא מעריכית יורדת, שמראה על סט נתונים לא מאוזן. חוסר אחידות זו משקפת את המציאות, שכן מגוון רחב של משחקים גורמת לקושי בהתעלות על שאר אלפי המשחקים ובהצלחה למכור כמות גדולה של מספר מיליוני מכירות.

* JP Sales

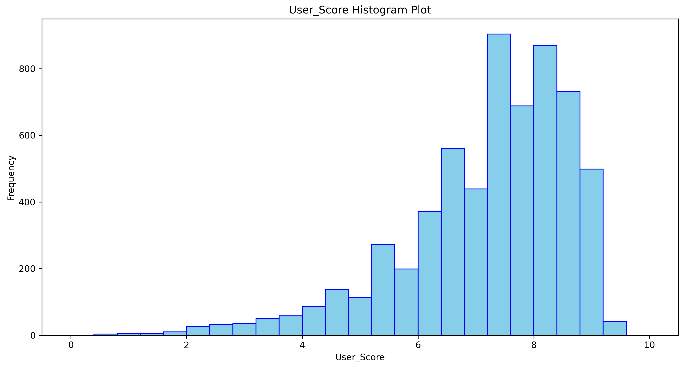
בדומה למשתנה הקודם, גם כאן אנו נתקלים במדד שמתייחס לכמות מכירות באזור גיאוגרפי מסוים, יפן. השוני בין יפן לאירופה מתבטא בהבדלי תרבויות שונים שגוררים התנהגות משתמשים אחרת וחיבה לתחביבים שונים, ועל כן איזור שונה זה עשוי להיות פיצ'ר שיתרום לנו להבין ולהסביר את המכירות באירופה. טווח הערכים הוא בין 0-6.5, כאשר כאן על פי ההיסטוגרמה ניכר לראות כי הרוב הגדול של המשחקים מדשדש במספר מכירות נמוך מאוד (רוב של משחקים שכלל לא נמכרו) וכאן ההתפלגות המעריכית בעלת הזנב הימני מתבטאת בצורה דרסטית ושמעידה על סט נתונים לא מאוזן.

* Other Sales

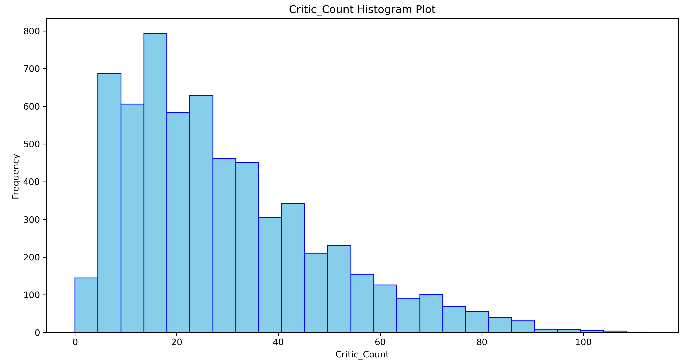
כפי שדנו בשני המשתנים הקודמים, גם מדד זה מתייחס למכירות שנעשו בייחוס למיקום גיאוגרפי, אך כאן מתייחס לשאר מדינות העולם. ההסתברות מלמדת מפאת המגוון הרחב של המדינות שפיצ'ר זה מאגד בתוכו בדומה למדד המכירות באירופה (בה עשרות מדינות) עשוי להטיב עם אימון המודל שלנו בהמשך, ולכן נבחן אותו בסעיפים הבאים. טווח הערכים נע בין 0-10.57 וגם על פי היסטוגרמה זו אנו מבחינים בגרף שאינו מאוזן בעל ירידה מעריכית (מתונה יותר מאשר יפן אך חדה יותר מאשר צפון אמריקה) כאשר מרבית המשחקים מכרו בין 0 ל0.2 מיליון.

* User Count

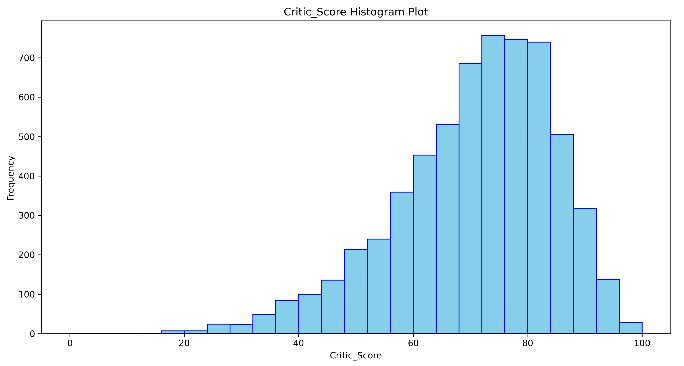
משתנה שסופר את כמות המשתמשים אשר העניקו דירוג. טווח הערכים הינו בין 4 ל10665 כאשר מרבית הערכים מתרכזים בתחילת ציר הX בין 0 ל200, כאשר ההתפלגות היא מעריכית יורדת והגרף אינו מאוזן. ההסתברות מלמדת על חוסר האחידות אך זה הגיוני ויכול לשקף את המציאות בה סביר שבשל המגוון הרב יהיה יותר משחקים בעלי כמות דירוגים נמוך, וככל שיעלה מספר הדירוגים כך ירד מספר המשחקים שדורגו כך.

* User Score:

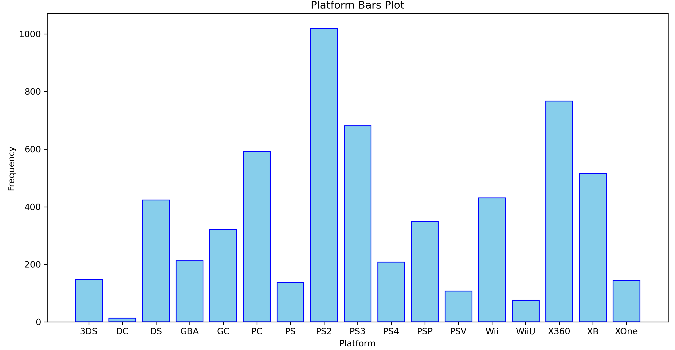
מדד זה למעשה מדבר על הציון הממוצע שאותו המשתמשים ברחבי העולם נתנו לכל משחק מחשב, כאשר ההיסטוגרמה מציגה את התדירות של כל ציון שניתן. ייתכן ומדד זה הינו מדד חשוב בעבודתנו מפני שהוא יכול להוות אינדיקציה טובה למכירות העתידיות של אותו משחק באירופה בעתיד. אנו יודעים שבמציאות כשמשתמש מחשב מעוניין לרכוש משחק מחשב, אחד המדדים הראשונים שאותו יבחן הוא כיצד משתמשים אחרים דירגו את המשחק הזה, וככל שהדירוג גבוה כך יגדלו הסיכויים שאחרים ירכשו אותו גם כן. מגרף זה אנו יכולים לראות את התפלגות הציונים שהביאו למשחקי המחשב שבבסיס הנתונים שלנו, כשאנו מבינים שהציון הממוצע נע סביב 7.2. טווח הערכים הוא בין 0.5 ל9.6 כשסט הנתונים אינו מאוזן ואחיד אלה יותר נראה כסוג של נורמלי עם זנב שמאלי.

* Critic count

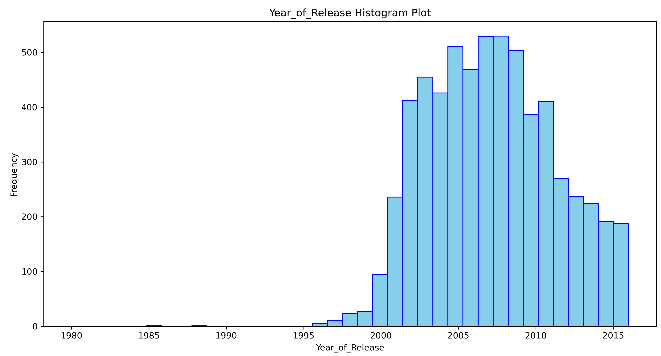
בדומה לפיצ'ר user count, גם כאן מדד זה מתייחס למספר המדרגים אך הפעם למבקרי המשחקים במקום לשחקנים עצמם. הטווח הוא בין 3 ל113, כאשר ההתפלגות הינה מעריכית יורדת (בעלת זנב ימני) ודבר זה עולה בקנה אחד עם המציאות בדומה לחוסר האחידות שהסברנו בפיצ'ר user count.

* Critic Score

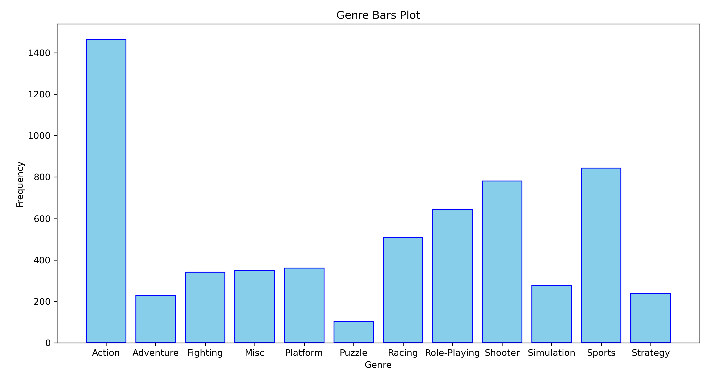
גם כאן בדומה לדירוג השחקנים ישנו פיצ'ר שמודד את הציון הממוצע שאותו המשתמשים ברחבי העולם נתנו לכל משחק מחשב, כאשר ההיסטוגרמה מציגה את התדירות של כל ציון שניתם. טווח הערכים הינו בין 13 ל98 כאשר ישנה התפלגות מעין נורמלית בעלת זנב ימני עם תוחלת של 70. לפי הגרף אנו רואים כי סט הנתונים לא מאוזן אך חוסר אחידות זו הגיונית, מסיבה דומה להסבר שניתן על ההסתברות בפיצ'ר User Score.

* Platform - קטגוריאלי

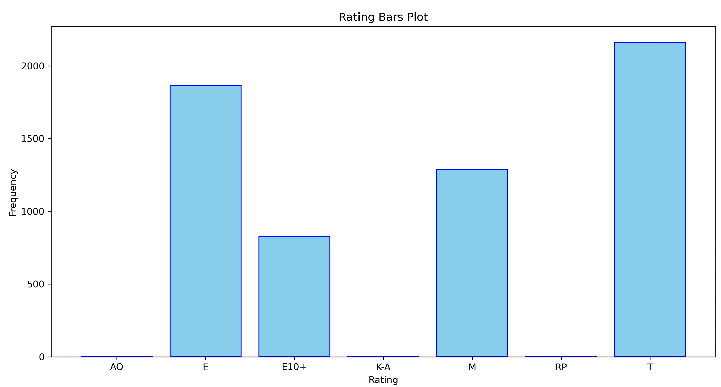
משתנה בדיד המייצג את הפלטפורמה עליה המשחק פועל כאשר ישנה השוואה כמותית בין הקטגוריות. מן ההיסטוגרמה אנו נחשפים לחוסר אחידות בין המכירות כתלות בסוגי הפלטפורמות השונים, כאשר הפערים בין הקטגוריות המובילות (PS2 ו X360 בראשם) אל מול אלה שפחות (כדוגמת DC) מלמדים אותנו על חוסר אחידות וסט נתונים שאינו מאוזן. אף על פי ריבויי האפשרויות לשחק במשחקים אלו, ישנן מספר פלטפורמות שמובילות את השוק ולכן חוסר אחידות הינה דבר הגיוני ומתיישר עם המציאות.

* Year of Release

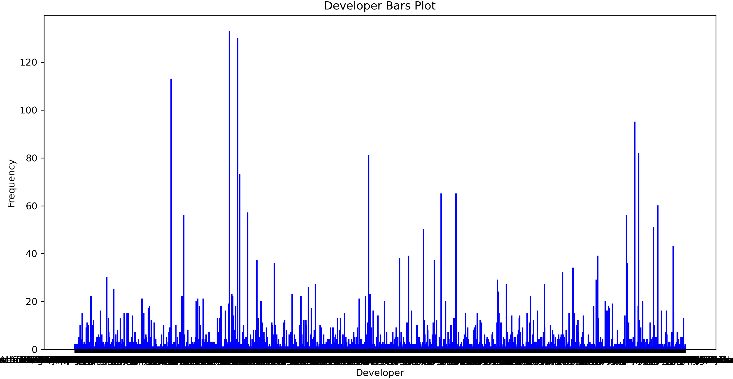
גרף זה מציג את המשחקים לפי סיווג של שנת יציאתם לאור. אנו ערים להתקדמות הטכנולוגיה עם השנים והפיתוחים באיכויות המשחק ושיפורים של פונקציונאליות למשתמש ולכן ניכר לראות מן ההיסטוגרמה את הגידול החד בין שנות 2000 ל כ2009 כששם הערך המקסימלי של כ500 משחקים, ומשם מתחילה ירידה כמעט בכל שנה לשנה העוקבת עד ל2016 (טווחי השנים הם 1985 עד 2016). הגרף אינו מאוזן ומראה עליה חדה ולאחריה ירידה מעט מתונה יותר עם השנים, וככלל לכן הוא אינו אחיד כפי שצפינו.

* Genre - קטגוריאלי

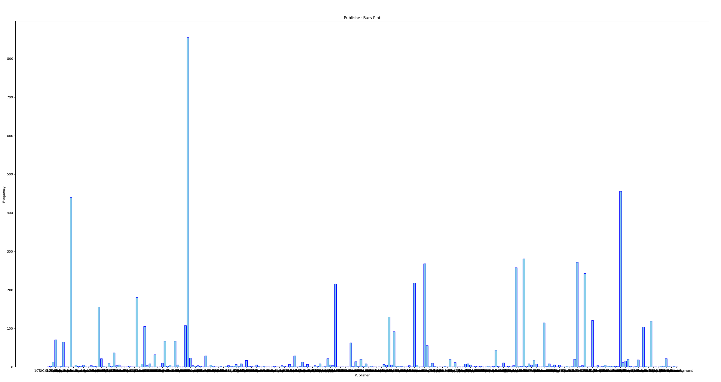
בגרף עמודות זה אנו רואים את הפיצ'ר שמייצג את סוג תוכן משחק המחשב. על פי ההיסטוגרמה, ישנה התפלגות שונה מאוד בין התכנים, כאשר קל להבחין בחוסר האחידות כדוגמת ריבוי קטגוריית action אל מול מיעוט ב puzzle וכדומה. התפלגות לא מאוזנת זו מתיישבת עם המציאות וההסתברות מלמדת על סוגי משחקים שיותר אהובים על ידי משתמשים לעומת כאלו אחרים ולכן דבר זה מתבטא בכמות המשחקים שייצרו מאותו תוכן. אם הרבה משתמשים בעולם מעידים על חיבתם לאקשן, אזי סביר להניח שחברות ייצרו יותר משחקים בתחום הזה מאשר השאר. כך אפשר להסביר את חוסר האיזון.

* Rating - קטגוריאלי

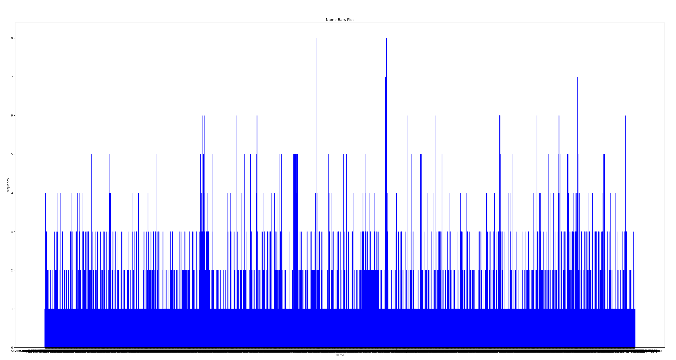
פיצ'ר זה מלמד על  דירוג תכנים למשחקי וידאו בארצות הברית ובקנדה, כאשר כל משחק מאופיין על ידי אחת מן 7 הקטגוריות. אנו רואים מן הגרף כי מתוך כל שבעת הקטגוריות, ישנה חוסר אחידות מובהקת כאשר רק 4 מתוכם משויכות למשחקים כמעט לחלוטין, ואילו גם ביניהן יש חוסר איזון. בשאר שלוש הקטגוריות ישנם מספר כמעט אפסי של משחקים, ובהמשך נבדוק מה נעשה עם דבר זה.

* Developer - קטגוריאלי

לפי ההיסטוגרמה של פיצ'ר זה המתאר את המפתח של כל משחק המחשב, אנו רואים פיזור גדול מאוד בכמות המפתחים, כאשר התדירות הגבוהה ביותר היא כ130, ואילו ישנם המון מפתחים בעלי משחקים בודדים. הטווח הוא בין 1 ל133, ואנו נראה בהמשך האם נתון זה הוא יטיב עם המודל או שמא עדיף להסיר.

* Publisher (קטגוריאלי)

משתנה זה מציג את מפרסם המשחק, וגם כאן אנו יכולים לראות כמות גדולה מאוד של קטגוריות שאין אנו יכולים ללמוד ממנה כמעט דבר, מלבד התפלגות לא אחידה וגרף מפוזר שנשקול בהמשך כיצד נתמודד עימו. הטווח הוא בין 1 ל855, ונתייחס בהמשך.

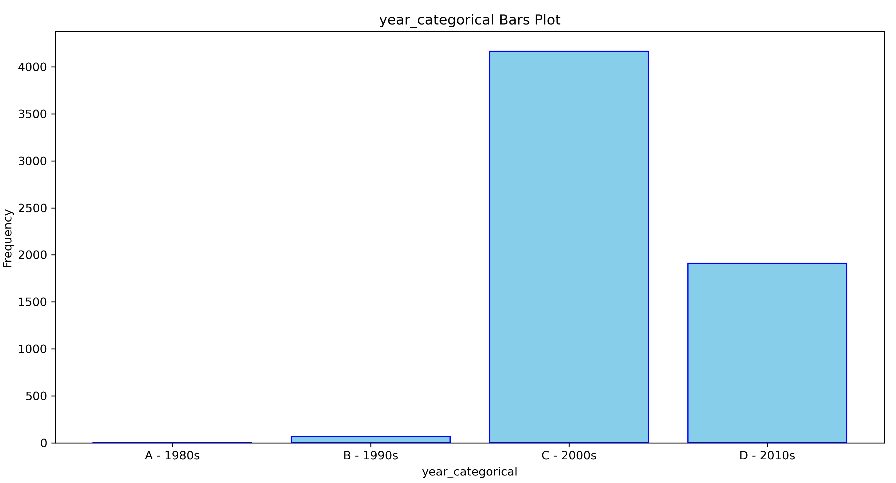
* Name (קטגוריאלי)

בדומה לשני המשתנים הקודמים, גם מהסתכלות על גרף עמודות זה אנו נחשפים לכמות גדולה מאוד של שמות המשחקים השונים, וגרף זה שמלמד על שאמנם מאוזן באופן יחסי יותר מהמשתנים האחרים, אך בהמשך נתון על טיבו והאם במהלך אימון המודל הוא תורם להצלחת התהליך.

**Data creation**

*Pre-processing:*

אנו נרצה לבצע מניפולציה על הדאטה אודות משחקי המחשב וזאת על מנת לשפר את הגישה והניתוח של הפיצ'רים השונים, על מנת שדבר זה יתרום בהמשך ללימוד אפקטיבי ויעיל של המודל שנבחר.

* ראשית, ביצענו **בדיקת יתירות** ווידאנו כי לא קיימת דופליקציה של ישויות, ואכן לא קיימות שורות כפולות.
* כאשר בדקנו אם קיימים **ערכים חסרים** בדאטה הבחנו ב"0" בחלק מהפיצ'רים- אך למדנו כי מדובר בנתון נומרי מאופס ולא בערך חסר. לדוגמה- למשחק "Cel Damage" שנתון כי בוצעו "0" מכירות ביפן, הדבר הגיוני. מעבר לכך – לא היו ערכים חסרים.
* **מעבר משתנים רציפים לקטגוריאליים** -בחנו אפשרות של החלת המרת סוג הדאטה עבור הפיצ'ר “Year\_of\_release” והפיכתו למשתנה קטגוריאלי אשר מבחין בין התקופות השונות וזאת על בסיס ידע קודם, זאת מכיוון שבמהלך העשורים האחרונים היו התפתחויות רבות בתחום משחקי המחשב – כמו יציאה של קונסולת PS2 בשנת 2000 אוWii בשנת 2006, אשר השפיעו על תפוצת המשחקים ואופיים. כך אנו סבורים שישנו היגיון לוגי נכון יותר בשינוי המכירות כתלות בעשור מאשר בשנה בודדת, שכן לעיתים קרובות למשחקים לוקח מספר שנים לפרוץ.  
  מכיוון שההשפעה של השינויים איננה מיידית וניתנת לצפייה החלטנו **לבצע חלוקת EWD אוטומטית**, על פי העשורים השונים.

כפי שאנו רואים מן הגרף הטווחים הינם :

* טווח A – שנים 1980 עד 1990.
* טווח B – שנים 1990 עד 2000.
* טווח C – שנים 2000 עד 2010.
* טווח D – שנים 2010 עד 2020.

כפי שציינו בסעיף הקודם, על אף שבחלק ניכר מן ההיסטוגרמות לא התקבל גרף מאוזן – מצאנו כי מרבית הנתונים אכן מתיישרים עם המצב במציאות שכן לעיתים קרובות הנתונים אכן אינם מאוזנים במציאות. סיבה נוספת לביצוע שינוי זה הינו מדד הפירסון הנמוך שיש למשתנה זה עם המשתנה המוסבר- מדד שנעשה בשלב מאוחר יותר של העבודה.

* **ניפוי חריגים:**

לאחר שבדקנו את התפלגות המשתנים האפריוריים, הוצאנו Box Plot מתאימים (פירוט תרשימי Box Plot בנספחים). ככלל, לא רצינו לבצע ניפוי חריגים שלא לצורך מכיוון שאנו סבורים שהמודל יוכל ללמוד גם מנתונים חריגים. במקרה זה ראינו כי ישנם נתונים חריגים רבים בפיצ'רים שונים – דבר שככל הנראה מעיד על השוק ולא נרצה לפספס תופעות אלה במודל.

אך עבור הרשומה של המשחק "Wii Sports" ראינו שמתקבלת ב"NA\_Sales" נתון שהוא באופן משמעותי קיצוני ביחס לשאר הנתונים- על אף שהוא איננו הנתון החריג היחיד העולה מן הגרפים.

לאחר שחקרנו את הנושא, ראינו שמשחק זה שווק יחד עם הקונסולה "Wii" שיצאה כאמור ב2006, על כן, נתון זה משקף תופעה שהינה מקרה פרטי ולמעשה משקפת את כמות המכירות של קונסולה ולא של משחק- על כן החלטנו לנפות רשומה זאת.

*Segmentation:*

תהליך טכני שמסייע להתמקד באלמנטים הרלוונטיים בישות, במקרה שלנו הדאטה אודות הישויות הובא כנתונים, נומריים וטקסטואליים, שאין צורך בביצוע סגמנטציה בהם.

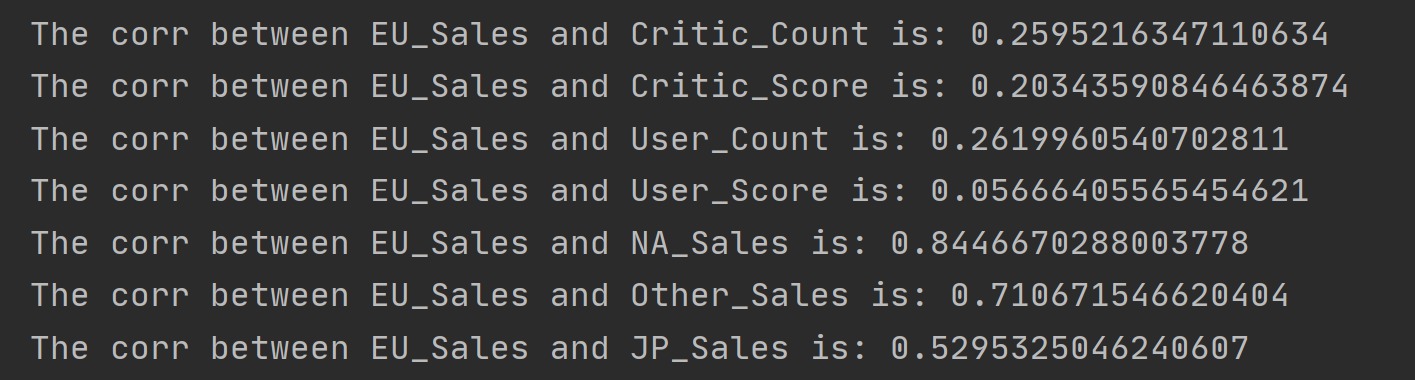
*Feature Extraction:*

תהליך חילוץ פיצ'רים מהData נועד לבצע פישוט (הורדת ממד) על ידי ייצוג יעיל יותר. התכונות הן למעשה המועמדות לייצוג הישות עבור המשימה הנדרשת. סוג המשימה יגזור את כמות וסוג הפיצ'רים המתאימים ביותר.

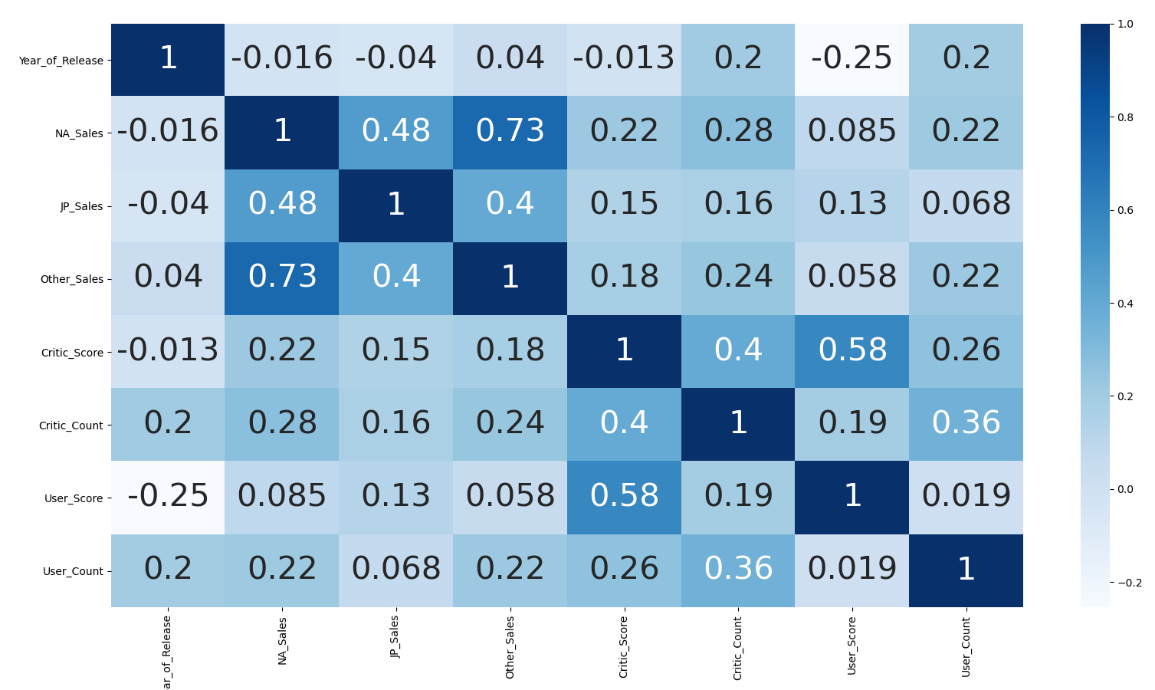
במקרה שלנו, מדובר בנתונים אשר אין להם יחס לזמן שהגיעו מחולצים. התכונות הן Fixed Number of Features וזהות עבור כל ישות -ומכאן שנשתמש ב- Knowledge Based Features.

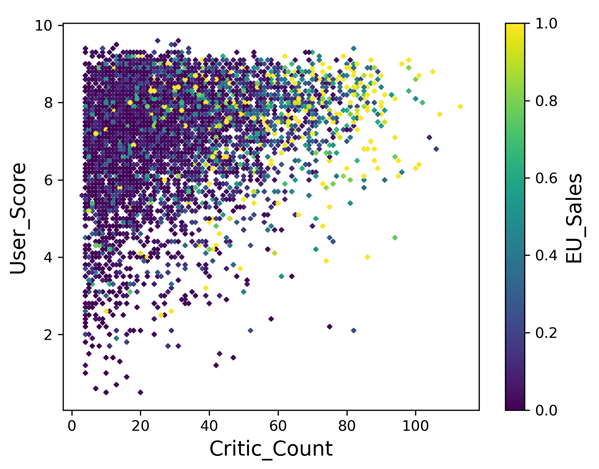
ניתן לראות כי הנתונים שהוצגו לנו נבחרו להיות מועמדים לייצוג הוגדרו ככל הנראה על ידי אדם שכן מספר הפיצ'רים אינו רב ומכיוון שמדובר בתכונות אינפורמטיביות.

את הורדת הממד ננסה ליצור על ידי בחירת זוגות פיצ'רים שנעשה להם קומבינציה.  
ראשית נרצה לבחון הן ויזואלית והן מספרית את מדד הפירסון של כל אחד מהפיצ'רים עם המשתנה המוסבר- מספר מכירות באירופה. להלן המדדים:



לאחר מכן נרצה לבדוק את הקורלציות בין הפיצ'רים גם כן על ידי מדד פירסון, להן הגרף:



נרצה למצוא פיצ'רים להם מדד פירסון נמוך ביחס למשתנה המוסבר, כמו כן, שאותם הפיצ'רים יהיו בעלי קורלציה נמוכה עם המשתנים האחרים- זאת במטרה לבצע איחוד משתנים, להביא למיצוי המידע ולדיסקרימינציה טובה יותר. הפיצ'רים Critic count ו- User score קיבלו את המדדים הנמוכים ביותר ביחס למשתנה המוסבר. בנוסף, הקורלציה ביניהם היא נמוכה באופן יחסית ועומדת על 0.19. נבצע מיזוג כך שבציר ה-X יהיה Critic count המתאר את מספר המבקרים שנתנו ביקורת למשחק, ובציר ה-Y את User score שהוא למעשה הדירוג הממוצע אותו קיבל המשחק מן המשתמשים. הצבעים בגרף מתארים את המשתנה המוסבר שהוא מספר המכירות באירופה. ניתן לראות ויזואלית כי קומבינציה זו אכן מתארת בצורה טובה יותר את המשתנה המוסבר, על כן נבצע הורד ממד על ידי מיזוג.

Feature Representation:

הנתונים אותם קיבלנו מיוצגים בחלקם על ידי הערכים הנומריים ובחלקם באמצעות המייצג משתנה קטגוריאלי או שם. בנוסף, קיימת קטגוריית "Reviewed" המיוצגת על ידי ערך בוליאני.  
לאחר בחינת הפיצ'רים, ראינו כי קיימות שתי עמודות נתונים אשר מייצגות את הציון אשר המשחק קיבל ממבקרים וממשתמשים. הסולם הינו שונה בשני הפיצ'רים על אף שהציון משקף את אותו טיב ההערכה. כאשר מדובר בדירוג המבקרים, המשתנים נעים בטווח 0-100 בעוד שעבור המשתמשים הטווח הינו בין 0-10. בכדי למנוע מצב בו נוצרת הטיה מפאת השוני בטווח הערכים, מצאנו לנכון לבצע Values Normalization לציונים ב-Scale שבין [0,1]. פעולה זו תגרום לאחידות בין הנתונים תוך שמירה על שונותם.

Feature Selection:

מטרת שלב זה הינה הסרת תכונות חסרות משמעות, שכן אלו עלולות להעמיס על תהליך העיבוד, לגרום להטעיות וכן להוביל לקורלציה משמעותית בין המשתנים (Features).

המטרה היא להגיד לסט פיצ'רים מצומצם וממצה ולהימנע מסיבוך יתר.

קיימות מספר אסטרטגיות שונות לבחירת הפיצ'רים. הן מתחלקות לרוב לאסטרטגיות כמותניות ואיכותניות.

הבחנו כי ישנו פיצ'ר חריג אשר מהווה משתנה חסר משמעות לחלוטין, המשתנה "Reviewed" הינו משתנה בינארי אשר מקבל ערך חיובי זהה עבור כלל הישויות. מכיוון שהפיצ'ר אינו תורם אינפורמציה למשימת הלימוד הוא מיותר- על כן החלטנו להסיר את פיצ'ר זה מבסיס הנתונים.

Dimensionality Reduction

בשלבים הקודמים אנו למעשה כבר ביצענו הן הסרה של פיצ'ר Reviewed שצפינו שלא יתרום ללימוד המודל והן איחוד פיצ'רים של Critic\_Count & User\_score , שכאמור תהליך זה גרם להורדת מימד., לכן כרגע איננו רואים עוד פיצ'ר נוסף שברצוננו להוריד. בנוסף, הLABLE בעבודה שלנו הינו רציף ולכן המטרה של הפרדה מקסימלית בין מחלקות אינו רלוונטי כרגע.

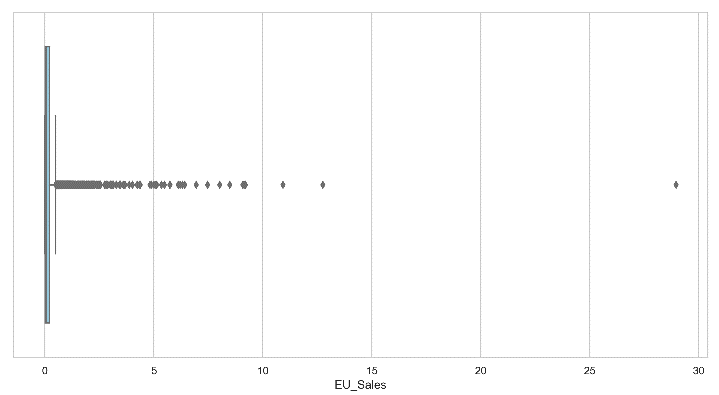
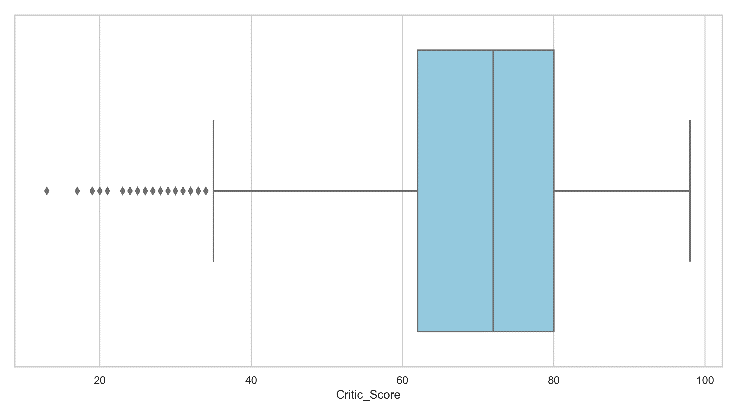
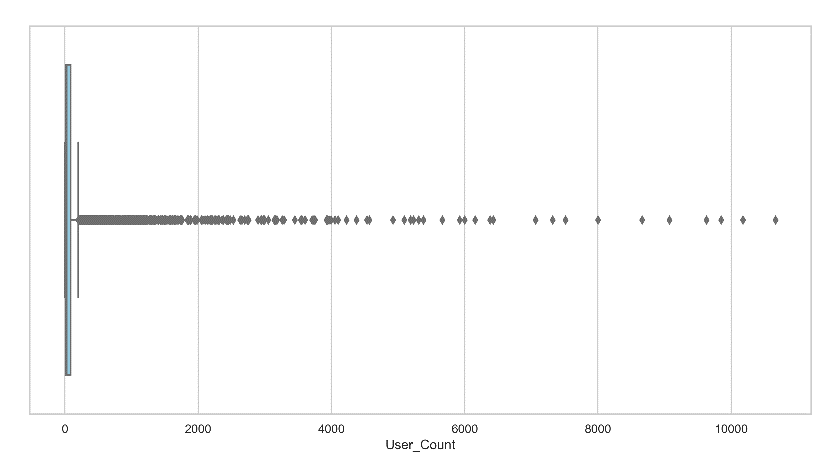
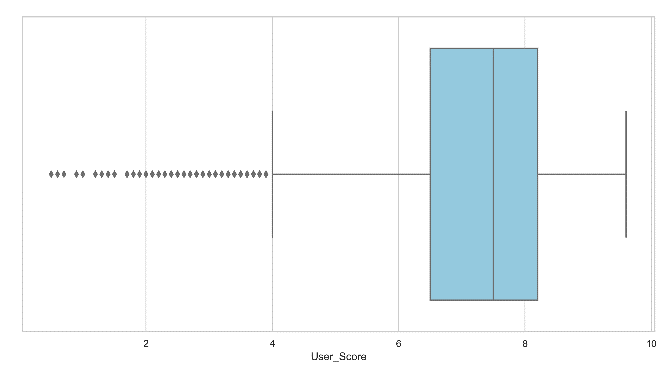
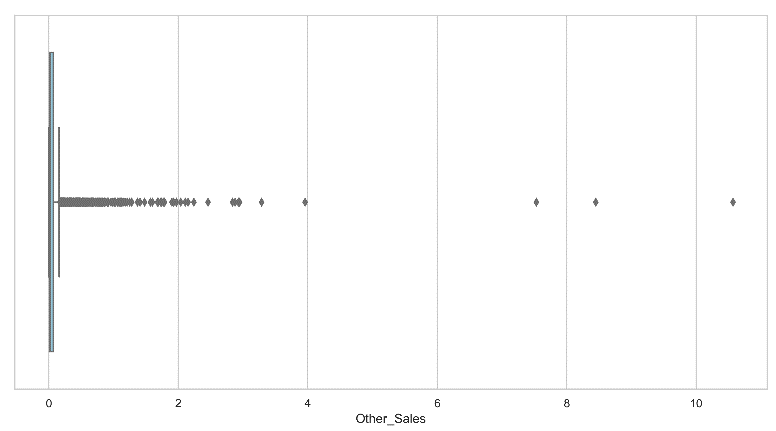
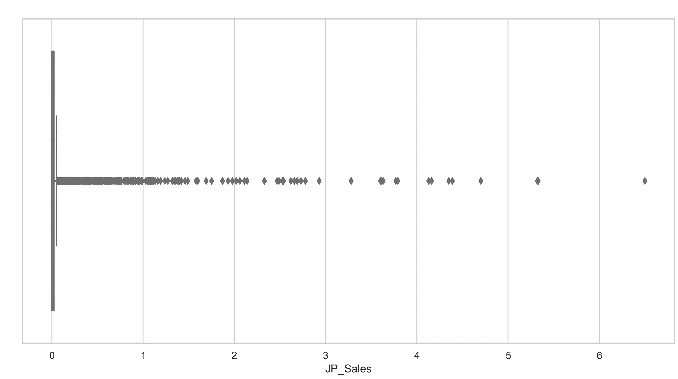
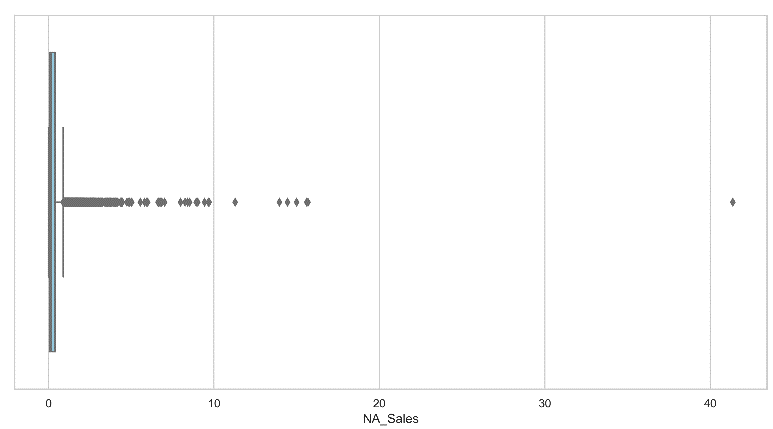
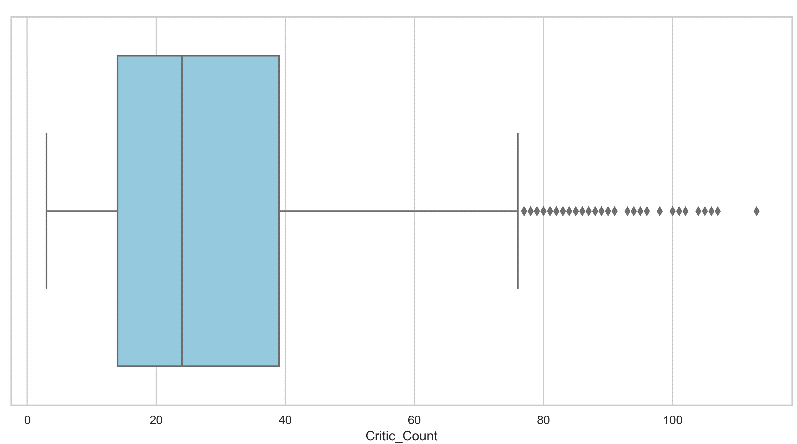
*Model Training:*

***ולידציה-***

את המודל אנו מאמנים ועושים induce על ידי training set. מכיוון שכמות המידע אותו אנו אוספים הוא סופי, לא נוכל באמת לדעת אם הוא מייצג את האוכלוסייה כולה.  
יש לנו בנוסף את test set, המייצג את העולם האמיתי, עליו אנחנו בוחנים את המודל בצורה "טהורה" שכן מדובר בדוגמאות שהמודל שלנו לא יודע עליהם (מכונה גם "סט אמת").  
בכדי להימנע ממצב של over fitting יש צורך בסט ולידציה, סט זה עוזר לנו להעריך את הפגמים במודל ובכך לקבל החלטה אם יש להשאיר את המודל או להחליפו, כמו כן, נוכל לבחון את בחירת הפרמטרים שנעשתה. בשלב הולידציה ניתן למעשה הערכה למודל שבנינו.   
נשתמש ב – cross validation על מנת להוריד את השונות בין validation sets השונים, בשלב זה נרצה לבדוק אם המידע שלנו תואם ומייצג את המתרחש במציאות.

בהינתן כמות הדגימות (מעל 6000 רשומות) ומשימת הלמידה- **נבחר בשיטת ה- K-fold** ובו נחלק את הTraining data ל-10 חבילות שונות ובכל איטרציה נבחר fold שישמש כtest set, נבצע 10 איטרציות על מנת לכסות את כל האפשרויות. בסוף התהליך נעריך את המודל על פי ממוצע השונות של הסטים.

נספחים:



תיעוד קוד:

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sn  
from itertools import product  
  
data\_x\_train = pd.read\_csv(r'C:\Users\karmo\PycharmProjects\ml11\X\_train.csv')  
data\_x\_test = pd.read\_csv(r'C:\Users\karmo\PycharmProjects\ml11\X\_test.csv')  
data\_y\_test = pd.read\_csv(r'C:\Users\karmo\PycharmProjects\ml11\y\_test\_example.csv')  
data\_y\_train = pd.read\_csv(r'C:\Users\karmo\PycharmProjects\ml11\y\_train.csv')  
  
  
def print\_hist(a):  
 plt.xlabel(a)  
 plt.ylabel('Frequency')  
 plt.title(a + ' Histogram Plot')  
 print('the max value of ' + a + ' is {}'.format(data\_x\_train[a].max()))  
 print('the min value of ' + a + ' is {}'.format(data\_x\_train[a].min()))  
 print('the mean value of ' + a + ' is {}'.format(data\_x\_train[a].mean()))  
 print()  
 plt.show()  
  
  
def print\_hist\_EU\_sales(a):  
 plt.xlabel(a)  
 plt.ylabel('Frequency')  
 plt.title(a + ' Histogram Plot')  
 print('the max value of' + str(a) + 'is {}'.format(data\_y\_train[a].max()))  
 print('the min value of' + str(a) + 'is {}'.format(data\_y\_train[a].min()))  
 print('the mean value of' + str(a) + 'is {}'.format(data\_y\_train[a].mean()))  
 plt.show()  
  
  
# histogram of continuous param  
  
######################### EU\_sales #############################  
plt.hist(data\_y\_train['EU\_Sales'], bins=50, range=(0,1), color="skyblue", edgecolor='blue', linewidth=1)  
print\_hist\_EU\_sales('EU\_Sales')  
  
plt.hist(data\_x\_train['User\_Score'], bins=25, range=(0, 10), color="skyblue", edgecolor='blue', linewidth=1)  
print\_hist('User\_Score')  
  
plt.hist(data\_x\_train['User\_Count'], bins=50, range=(0,200), color="skyblue", edgecolor='blue', linewidth=1)  
print\_hist('User\_Count')  
  
plt.hist(data\_x\_train['Critic\_Count'], bins=25, range=(0,113), color="skyblue", edgecolor='blue', linewidth=1)  
print\_hist('Critic\_Count')  
  
plt.hist(data\_x\_train['Critic\_Score'], bins=25, range=(0,100), color="skyblue", edgecolor='blue', linewidth=1)  
print\_hist('Critic\_Score')  
  
plt.hist(data\_x\_train['Critic\_Score'], bins=25, range=(0,100), color="skyblue", edgecolor='blue', linewidth=1)  
print\_hist('Critic\_Score')  
  
plt.hist(data\_x\_train['NA\_Sales'], bins=50, range=(0,4), color="skyblue", edgecolor='blue', linewidth=1)  
print\_hist('NA\_Sales')  
  
plt.hist(data\_x\_train['JP\_Sales'], bins=50, range=(0,0.5), color="skyblue", edgecolor='blue', linewidth=1)  
print\_hist('JP\_Sales')  
  
plt.hist(data\_x\_train['Other\_Sales'], bins=50, range=(0,1), color="skyblue", edgecolor='blue', linewidth=1)  
print\_hist('Other\_Sales')  
  
plt.hist(data\_x\_train['Year\_of\_Release'], bins=37, range=(1980,2016), color="skyblue", edgecolor='blue', linewidth=1)  
print\_hist('Year\_of\_Release')  
  
  
# histogram of discrete param  
  
def bars(a):  
 plt.xlabel(a)  
 plt.ylabel('Frequency')  
 plt.title(a + ' Bars Plot')  
 keys, counts = np.unique(data\_x\_train[a], return\_counts=True)  
 plt.bar(keys, counts, color="skyblue", edgecolor='blue', linewidth=1)  
 plt.show()  
  
  
bars('Genre')  
bars('Rating')  
bars('Developer')  
bars('Platform')  
bars('Publisher')  
  
  
---------------------- continuous to categorized fichers----------------------#  
  
def get\_decay\_from\_year(year):  
 if 1980 <= year < 1990:  
 return 'A - 1980s'  
 elif 1990 <= year < 200:  
 return 'B - 1990s'  
 elif 2000 <= year < 2010:  
 return 'C - 2000s'  
 else:  
 return 'D - 2010s'  
  
  
  
  
data\_x\_train['year\_categorical'] = data\_x\_train['Year\_of\_Release'].apply(get\_decay\_from\_year)  
bars('year\_categorical')  
  
---------------------- BoxPlots----------------------#  
  
sn.set\_theme(style="whitegrid")  
sn.boxplot(data=data\_x\_train, x = 'User\_Score' , color ='skyblue')  
plt.show()  
sn.boxplot(data=data\_x\_train, x = 'User\_Count', color ='skyblue')  
plt.show()  
sn.boxplot(data=data\_x\_train, x = 'Critic\_Score' , color ='skyblue')  
plt.show()  
sn.boxplot(data=data\_x\_train, x = 'Critic\_Count' , color ='skyblue')  
plt.show()  
sn.boxplot(data=data\_x\_train, x = 'NA\_Sales' , color ='skyblue')  
plt.show()  
sn.boxplot(data=data\_x\_train, x = 'JP\_Sales' , color ='skyblue')  
plt.show()  
sn.boxplot(data=data\_x\_train, x = 'Other\_Sales' , color ='skyblue')  
plt.show()  
sn.boxplot(data=data\_y\_train, x = 'EU\_Sales' , color ='skyblue')  
plt.show()  
  
  
---------------------- remove row ----------------------#  
data\_x\_train.drop(data\_x\_train.index[2951])  
data\_x\_train.drop[data\_x\_train.name = 'Wii Sports']  
print(data\_x\_train.loc[2951])  
print(data\_x\_train.drop(data\_x\_train.index[2951]))  
print(data\_x\_train.loc[2951])  
  
---------------------- correlation matrix ----------------------#  
corr\_matrix = data\_x\_train.corr()  
sn.heatmap(corr\_matrix, annot=True, cmap="Blues", annot\_kws={"size": 30})  
plt.show()  
  
---------------------- count results ----------------------#  
data\_x\_train[['Genre']].groupby('Genre').count()  
a = (data\_x\_train[['Genre', 'Name']].groupby('Genre').count())/(6142)  
# data\_x\_train[['Genre', 'User\_Score']].groupby('Genre').sum()  
  
---------------------- correlation\_x\_y ----------------------#  
def corr\_x\_y(a):  
 # plt.xlabel(a)  
 # plt.ylabel('EU\_Sales')  
 # plt.title(a + ' and EU\_Sales Plot')  
 # plt.plot(data\_x\_train[a], data\_y\_train['EU\_Sales'], '.', color='darkblue')  
 # plt.show()  
  
for i in data\_x\_train.columns:  
 corr\_x\_y(i)  
  
---------------------- corr\_x\_y value ----------------------#  
print('The corr between EU\_Sales and Critic\_Count is: ' + str(data\_x\_train['Critic\_Count'].corr(data\_y\_train['EU\_Sales'])))  
print('The corr between EU\_Sales and Critic\_Score is: ' + str(data\_x\_train['Critic\_Score'].corr(data\_y\_train['EU\_Sales'])))  
print('The corr between EU\_Sales and User\_Count is: ' + str(data\_x\_train['User\_Count'].corr(data\_y\_train['EU\_Sales'])))  
print('The corr between EU\_Sales and User\_Score is: ' + str(data\_x\_train['User\_Score'].corr(data\_y\_train['EU\_Sales'])))  
print('The corr between EU\_Sales and NA\_Sales is: ' + str(data\_x\_train['NA\_Sales'].corr(data\_y\_train['EU\_Sales'])))  
print('The corr between EU\_Sales and Other\_Sales is: ' + str(data\_x\_train['Other\_Sales'].corr(data\_y\_train['EU\_Sales'])))  
print('The corr between EU\_Sales and JP\_Sales is: ' + str(data\_x\_train['JP\_Sales'].corr(data\_y\_train['EU\_Sales'])))  
  
---------------------- combine 2 categories & EU\_Sales\_heatmap ----------------------#  
def Change\_Values(val):  
 if val > 1:  
 return 1  
 else:  
 return val  
  
  
data\_y\_train['EU\_Sales\_heatmap'] = data\_y\_train['EU\_Sales'].apply(Change\_Values)  
plt.scatter(data\_x\_train['Critic\_Count'], data\_x\_train["User\_Score"], c=data\_y\_train['EU\_Sales\_heatmap'], s=4,  
 marker='o')  
plt.colorbar().set\_label('EU\_Sales\_range', fontsize=14)  
plt.xlabel('Critic\_Count', fontsize=14)  
plt.ylabel('User\_Score', fontsize=14)  
plt.title('EU\_Sales bias Critic\_Count & User\_Score ', fontsize=17)  
plt.show()  
  
  
---------------------- Feature Representation: ----------------------#  
def Change\_Score\_Value\_Critic(val):  
 return val / 100  
  
  
def Change\_Score\_Value\_User(val):  
 return val / 10  
  
  
data\_x\_train['Critic\_Score\_New'] = data\_x\_train['Critic\_Score'].apply(Change\_Score\_Value\_Critic)  
data\_x\_train['User\_Score\_New'] = data\_x\_train['User\_Score'].apply(Change\_Score\_Value\_User)  
data\_x\_train = data\_x\_train.drop(columns=['Critic\_Score','User\_Score'])  
print()  
---------------------- remove Exceptions ----------------------#  
data\_x\_train.drop([2951])  
  
data\_x\_train = data\_x\_train[data\_x\_train.Name != "Wii Sports"]  
data\_x\_train = data\_x\_train.drop(columns=['Name', 'Reviewed'])  
print()  
  
plt.plot(data\_x\_train['NA\_Sales'], data\_x\_train['Other\_Sales'], '.', color='darkblue')  
plt.ylim((0,4))  
plt.xlim((0,15))  
plt.show()