**אלגוריתם השכן הקרוב (k-Nearest Neighbors - k-NN)**

KNN הוא אלגוריתם חסר פרמטרים, כלומר הוא לא "לומד" נוסחה קבועה או קשר מוגדר בין התכונות, אלא שומר את כל הדוגמאות ובזמן אמת מזהה את השכנים הקרובים ביותר לצורך חיזוי. האלגוריתם משמש ללמידת מכונה ומתאים גם לסיווג (Classification) וגם לרגרסיה (Regression).

איך פועל KNN לרגרסיה?

1. בהינתן נקודת קלט חדשה, האלגוריתם מזהה את k השכנים הקרובים ביותר במרחב הנתונים.
2. ערך החיזוי מחושב כממוצע של ערכי השכנים הללו.
3. ניתן לשקלל את תרומת השכנים כך ששכנים קרובים יותר ישפיעו יותר מהרחוקים.
4. שיטת שקלול נפוצה היא שימוש במרחק הפוך, בה כל שכן מקבל משקל של 1/d (כאשר d הוא המרחק בין נקודת הקלט לשכן).
5. שיטה זו מפחיתה את השפעת נקודות רחוקות שעלולות להיות פחות רלוונטיות לחיזוי.

KNN ממפה את הדאטה למרחב n-מימדי

האלגוריתם פועל במרחב רב-ממדי, שבו כל תצפית מתוארת על ידי מספר תכונות.

על ידי חישוב המרחק בין נקודות במרחב זה, ניתן לקבוע את k השכנים הקרובים ביותר ולחשב את החיזוי על בסיסם.

**מדדי מרחק אפשריים (Distance Metrics)**

מכיוון שהאלגוריתם מבוסס על חישוב קרבה בין דוגמאות, בחירת מדד המרחק חשובה מאוד:

* מרחק אוקלידי (Euclidean Distance) – המרחק הגיאומטרי הרגיל בין שתי נקודות. זהו המדד הנפוץ ביותר.
* מרחק מנהטן (Manhattan Distance) – סכום ההפרשים המוחלטים בין התכונות. מתאים במיוחד כאשר הנתונים בנויים במבנה רשת (כמו רחובות בערים).
* מרחק משוקלל (Weighted Distance) – כל שכן מקבל משקל בהתאם למרחק שלו מנקודת הקלט – שכנים קרובים יותר יקבלו משקל גבוה יותר.

**בחירת K – מספר השכנים הקרובים**

* K קטן מדי → המודל יהיה רגיש מאוד לרעש ולערכים חריגים, מה שעלול לגרום ל-Overfitting (התאמת יתר).

אם K=1, התחזית תהיה בדיוק כמו ערך הנקודה הקרובה ביותר, מה שיגרום לרגישות גבוהה מאוד לרעש.

* K גדול מדי → המודל יחליק את הנתונים יתר על המידה, ייבצע ממוצע רחב מדי ויאבד דיוק (Underfitting).

אם K=10, התחזית תתבסס על ממוצע של 10 הנקודות הקרובות, מה שעשוי להפחית את השפעת הרעש.

לכן, יש למצוא ערך K אופטימלי כדי לאזן בין שני המצבים – לרוב נעשה זאת באמצעות Cross-Validation.

חשיבות התאמות ידניות (Tweaking) באלגוריתם

ניתן לכוונן את האלגוריתם על ידי בחירת מספר ה-K המתאים, סוג מדד המרחק (אוקלידי, מנהטן וכו'), ואופן חישוב המשקלות. לדוגמה, ניתן לבחור אם לכל שכן יהיה משקל שווה בממוצע, או אם השכנים הקרובים יותר יקבלו השפעה גבוהה יותר על החיזוי.

**צורך ב:Scaling / Normalizing**

מכיוון שאלגוריתם KNN מבוסס על חישוב מרחקים, יש לבצע נירמול (Normalization) או סטנדרטיזציה (Standardization) כדי למנוע השפעה לא הוגנת של משתנים עם טווחי ערכים שונים.

ללא Scaling, תכונות עם ערכים גדולים (כגון מחירים גבוהים) ישלטו על חישובי המרחק ויטו את התחזיות. שימוש ב-StandardScaler מבטיח שהנתונים יהיו בעלי ממוצע 0 ושונות 1, כך שכל התכונות יקבלו משקל שווה ולא ישפיעו באופן לא פרופורציונלי על המרחקים.

**יתרונות מול חסרונות של KNN לרגרסיה**

יתרונות

* מודל פשוט וקל להבנה – אינו דורש הנחות מורכבות על הנתונים.
* ניתן לבצע התאמות מרובות (K, מרחקים, משקולות) כדי לשפר ביצועים.
* אינו דורש אימון ממושך – אין שלב למידה, אלא חישוב ישיר של מרחקים.

חסרונות

* לא מתאים לדאטה גדול – חישוב מרחקים בין כל הנקודות לכל דגימה חדשה הוא איטי מאוד (סיבוכיות של O(n⋅d)).
* רגיש מאוד לערכים חריגים (Outliers) – בחירה לא נכונה של K עלולה לגרום לרגישות לרעש.
* תלוי בבחירת המרחק והנירמול – ביצועים משתנים מאוד בהתאם למדד המרחק והשיטה שנבחרה ל-Scaling.

תהליך העבודה במחברת לצורך חיזוי מחירי טיסה באמצעות KNN:

* הרצת המודל הראשוני של KNN עם הגדות ברירת המחדל
* כוונון פרמטרים באמצעות GridSearchCV –
* בחינת ערכים שונים של K כדי למצוא את מספר השכנים האופטימלי.
* בחינת מדדי מרחק שונים (אוקלידי, מנהטן) והשפעתם על הדיוק.
* בדיקה האם השימוש במשקלים (weights=distance) משפר ביצועים.
* שימוש ב-GridSearchCV כדי לבדוק 140 שילובים שונים ולבחור את הטוב ביותר.
* הרצת המודל עם הפרמטרים האופטימליים - K=15, מרחק מנהטן, ומשקל מבוסס מרחק.