**עץ החלטה לרגרסיה**

עץ החלטה הוא אלגוריתם למידת מכונה שניתן להשתמש בו בבעיות סיווג (Classification) ובבעיות רגרסיה (Regression). במקרה שלנו, אנחנו מתמקדים בעץ החלטה לרגרסיה, המשמש לחיזוי ערכים רציפים (חיזוי מחירים של טיסות בהתבסס על תכונות שונות).

עץ החלטה לרגרסיה לא מניח קשר לינארי בין הפיצ'רים למטרה, ובכך הוא מאפשר גמישות בחיזוי ומסוגל להתמודד עם נתונים בעלי קשרים מורכבים.

איך עץ החלטה לרגרסיה עובד

* חלוקה של הנתונים: עץ החלטה מחלק את הנתונים בכל צומת (Node) על פי ערכים של תכונות הקלט (Features). המטרה היא לחלק את הנתונים כך שהשונות (variance) בין הערכים בקבוצות שנוצרות תהיה מינימלית. בכל צומת, האלגוריתם בוחר את התכונה ואת הערך שמקטינים בצורה המיטבית את השונות בקבוצות.
* תחזית לפי ממוצע: כאשר העץ מגיע לעלים (Leaves), התחזית היא ממוצע הערכים של הדוגמאות בתוך אותה קבוצה. למשל, אם העלה מכיל טיסות עם מחירים שונים, התחזית תהיה ממוצע המחירים בטיסות אלה.
* קריטריון לפיצול: הקריטריון לפיצול בעץ החלטה לרגרסיה הוא הקטנת השונות (Variance Reduction).

יתרונות וחסרונות של עץ החלטה לבעיות רגרסיה:

יתרונות:

* פשטות והבנה: עץ החלטה קל לפירוש, וניתן להציג את תהליך החיזוי בצורה גרפית.
* יכולת טיפול בקשרים לא לינאריים: העץ מבצע חלוקות חוזרות של הנתונים בצמתים כך שכל חלוקה מבוססת על תכונה שונה, מבלי להניח קשר לינארי בין התכונות.
* עבודה עם נתונים קטגוריאליים ומספריים: העץ מבצע חלוקה של הנתונים בצורה שתמזער את השונות בקבוצות, ללא קשר לסוג הנתון.
* מעט עיבוד נתונים מקדים: אין צורך בנרמול או התאמת ערכים לסקאלה אחידה, כי כל תכונה נבדקת בנפרד.
* פחות רגיש לערכים חריגים: ערכים חריגים לא בהכרח יגרמו לעץ להתאים את עצמו בצורה לא נכונה, כי ההחלטות מתקבלות לפי תכונות ספציפיות בכל צומת.
* יכולת להתמודד עם נתונים חסרים: המודל יכול להתמודד עם ערכים חסרים על ידי שימוש בהשוואות בין הערכים הקיימים.

חסרונות:

* נטייה ל-Overfitting: אם העץ עמוק מדי, הוא יכול להתאים את עצמו טוב מדי לנתוני האימון ולפגום ביכולת הכללה על נתונים חדשים.
* רגישות לרעש: עץ החלטה יכול להיות רגיש מאוד לרעש או תכונות לא רלוונטיות, ולהתאים את עצמו לניואנסים או שגיאות קטנות בנתונים, דבר שמוביל לחיזוי לא מדויק על נתונים חדשים.
* שונות גבוהה בתוך עלים: אם בתוך עלה יש שונות גבוהה בין הערכים, תחזיות המודל עשויות להיות לא מדויקות כי המודל מתבסס על ממוצע הערכים בקבוצה.
* חוסר יציבות: שינוי קטן בנתונים יכול להוביל לשינוי משמעותי במבנה העץ, מה שעלול לגרום לחוסר יציבות ואמינות של המודל.
* חוזק נמוך יחסית לעצים מרובים: עץ החלטה בודד פחות חזק ממודלים כמו Random Forest, שמשתמשים בכמה עצים כדי לשפר את הדיוק.

**שלבים במחברת של עץ החלטה:**

1. ניתוח קורלציות

בשלב הראשון ביצענו ניתוח קורלציות (רק על תכונות מספריות בעזרת select\_dtypes(include=['number'])) כדי להבין את הקשר בין מחיר הטיסה (Price(NIS)) לפיצ'רים נוספים. הצעד הזה עוזר לנו לזהות אילו תכונות קשורות למחיר בצורה חזקה, ולסנן תכונות לא רלוונטיות לפני שאנו מתחילים לבנות את המודל. לקחנו את 10 המשתנים עם הקורלציה (בערך מוחלט) הגבוהה ביותר.

1. פיצול הנתונים

לפני שמאמנים את המודל, אנחנו מגדירים את התכונות (X) ואת משתנה היעד (y- מחיר הטיסה). מחלקים את הנתונים לסט אימון (70%) וסט בדיקה (30%). תהליך זה מבטיח שהמודל יאומן על סט אחד של נתונים, והביצועים שלו ייבדקו על סט נפרד, דבר שמדמה איך הוא יתפקד על נתונים חדשים שלא ראה.

1. אימון המודל הראשוני והערכת ביצועים

השתמשנו ב- DecisionTreeRegressor לאימון המודל על נתוני האימון (X\_train, y\_train). המודל מאמן את עצמו על פי נתונים אלו כדי ללמוד את הקשר בין תכונות (features) למחיר (target).

בשלב זה מודל אומן תחילה עם פרמטרים ברירת מחדל של ה-DecisionTreeRegressor.

לאחר שהמודל אומן, השתמשנו ב- predict() כדי לבצע תחזיות על סט הנתונים של הבדיקה (X\_test). המודל מנסה לחזות את מחירי הטיסות לפי תכונות שהיו בסט הבדיקה.

**מדדים להערכת הביצועים- בהקשר של עץ החלטה:**

* MSE (Mean Squared Error): MSE גבוה מצביע על overfitting, כלומר המודל מתאים את עצמו בצורה מדויקת מדי לנתוני האימון ולא מצליח להכליל כראוי על נתונים חדשים. זה יכול להיגרם כאשר העץ עמוק מדי או שיש יותר מדי פיצולים, מה שמוביל למודל שמגיב בצורה יתרה לרעש בנתונים.
* MAE (Mean Absolute Error): MAE מודד את הממוצע של השגיאות האבסולוטיות (ההפרש המוחלט בין הערך הצפוי לערך האמיתי). אם ה-MAE גבוה, זה מעיד על טעות תחזית גדולה יותר.

MAE נמוך מציין שמודל מצליח לחזות בצורה מדויקת, עם טעויות תחזיות נמוכות.

* RMSE (Root Mean Squared Error): RMSE הוא שורש הריבועי של ה-MSE ומודד את הפער בין הערכים הצפויים והערכים האמיתיים בצורה רגיש יותר לשגיאות גדולות. RMSE גבוה מציין שהמודל לא מתאים בצורה מדויקת מאוד לנתונים, ויכול להצביע על overfitting במודל.
* RMSE נמוך מציין התאמה טובה בין התחזיות והערכים האמיתיים.
* R² Score: R² נמוך מציין שמודל לא מצליח להסביר את השונות בנתונים, כלומר התחזיות רחוקות מהערכים האמיתיים. R² גבוה מצביע על מודל שמסביר חלק גדול מהשונות בנתונים, כלומר התחזיות מדויקות יותר.

# המדדים מחושבים על y\_test (הערכים האמיתיים בסט הבדיקה) ו-y\_pred (התחזיות של המודל) כדי להעריך את ביצועי המודל על נתונים שלא ראה בזמן האימון. זה מאפשר לוודא שהמודל מסוגל להכליל ולהצליח על נתונים חדשים.

1. Feature Importance (חשיבות המאפיינים)

בשלב הזה ביצענו ניתוח של חשיבות המאפיינים כדי להבין אילו תכונות משפיעות יותר על חיזוי מחירי הטיסות. זיהינו את המאפיין Departure Month ככזה שיש לו השפעה נמוכה על המודל, ולכן החלטנו להסיר אותו. לאחר מכן, התרכזנו במאפיינים החשובים יותר, וביצענו אימון מחדש של המודל עם המאפיינים המשמעותיים בלבד, במטרה לשפר את דיוק המודל ולפשט אותו.

המסקנה שלנו היא שהסרת המאפיינים בעלי החשיבות הנמוכה לא השפיעה בצורה משמעותית על ביצועי המודל. ה-MSE וה-R² נשארו כמעט זהים, מה שמעיד שהמאפיינים שהוסרו לא היו חשובים לחיזוי המחירים.

1. תיאום פרמטרים (Hyperparameter Tuning)

שלב שבו אנו משתמשים בטכניקות כמו GridSearchCV על מנת למצוא את הערכים האופטימליים של הפרמטרים השונים במודל, במטרה לשפר את ביצועי המודל. הגדרנו רשת של ערכים אפשריים עבור פרמטרים:

* max\_depth: מגביל את עומק העץ כדי למנוע ממנו להיות עמוק מדי וללמוד יותר מדי פרטים לא חשובים.
* min\_samples\_split: קובע את מספר הדוגמאות המינימלי הנדרש כדי לפצל צומת, ומונע יצירת צמתים עם נתונים לא מספיקים.
* min\_samples\_leaf: קובע את מספר הדוגמאות המינימלי הנדרש בכל עלה, כדי להימנע מיצירת עלים עם מעט דוגמאות שלא מספקים תחזיות יציבות.

GridSearchCV מבצע קרוס-וולידציה (במקרה הזה 5 קיפולים) על כל קומבינציה של פרמטרים ובוחר את הקומבינציה הטובה ביותר.

אחרי שהתאמנו את המודל עם הפרמטרים הטובים ביותר שנמצאו, הערכנו שוב את ביצועי המודל באמצעות אותם מדדים וראינו שיפור (מה שמעיד על כך שהמודל הפך ליותר מדויק ומפוקס לאחר הרגולריזציה).

הפרמטרים הכי טובים שנמצא עבור מודל Decision Tree:

* criterion='poisson': קריטריון למדידת איכות הפיצול בצומת (כך שהשונות בתוך כל קבוצה תהיה קטנה יותר). Poisson הוא קריטריון המשמש בעיקר כשעובדים עם נתונים שהם קשר בין משתנים תלויים.
* max\_depth=None: המשמעות היא שאין הגבלה על עומק העץ. כלומר, העץ יכול להמשיך להתפצל עד שהוא לא יכול להתפצל יותר, ללא מגבלה על מספר הצמתים או העומק. הגדרה זו מאפשרת לעץ ללמוד בצורה מאוד מדויקת את הנתונים, אך עשויה להוביל ל-overfitting אם הנתונים מאוד שונים ומגוונים.
* min\_samples\_leaf=1: זהו המספר המינימלי של דוגמאות שנדרשות בכל עלה (leaf) בעץ. כשיש ערך של 1 כאן, זה מאפשר לכל עלה לכלול רק דוגמה אחת, מה שעשוי לגרום להתאמה מדויקת מדי של המודל לנתוני האימון (overfitting).
* min\_samples\_split=10: זהו המספר המינימלי של דוגמאות שנדרשות כדי לפצל צומת בעץ. עם ערך של 10, דרוש לפחות 10 דוגמאות בכל צומת כדי שהוא יתפצל. זה עוזר למנוע יצירת צמתים שמתפצלים רק על בסיס דוגמאות מעטות, ומסייע במניעת overfitting.

לאחר שלב זה בדקנו והדפסנו:

1. השוואה בין המחירים האמיתיים למחירים שהמודל חזה בסט הבדיקה, וכך אפשר להעריך את ביצועי המודל.
2. Residual Plot - בגרף זה אנו רואים את ההבדלים בין המחירים האמיתיים למחירים החזויים, כלומר את השגיאות (Actual - Predicted). רוב השגיאות נמצאות סביב 0, מה שמעיד שאין הטיית תחזיות ברורה.

ככל שהמחיר החזוי גבוה יותר, השגיאות נוטות להיות גדולות יותר, מה שמעיד שהמודל מתקשה לחזות בצורה מדויקת טיסות יקרות יותר.

1. Actual vs. Predicted Prices - גרף זה משווה בין המחירים האמיתיים למחירים החזויים.

רוב הנקודות קרובות לקו המנוגד (הקו האדום), מה שמעיד שהתחזיות בדרך כלל די מדויקות.

השגיאות גדולות יותר בטיסות יקרות, מה שמעיד שהמודל מצליח פחות לחזות את המחירים של טיסות יקרות.

1. ניסיון לשיפור המודל באמצעות מעבר לRandom Forest

המודל של Random Forest מציג שיפור ב-MSE ו-R², כלומר הוא מבצע חיזויים מדויקים יותר ומסביר יותר מהשונות במחירים בהשוואה ל-Decision Tree. החיזויים פחות נוטים לטעויות גדולות, והמגוון שבין העצים ב-Random Forest עוזר להפחית את ה-overfitting.

**Ensemble (שילוב מודלים):**

טכניקה בלמידת מכונה שמשלבת כמה מודלים ("לומדים") כדי לשפר את הביצועים של תחזיות. הרעיון המרכזי הוא שלכל מודל יש את המגבלות והחוזקות שלו, ולכן שילוב של מספר מודלים יכול להוביל לתוצאות מדויקות ויציבות יותר. למדנו על 2 שיטות:

* Bagging - טכניקת Ensemble שמטרתה להפחית את השונות (variance) במודל ולמנוע overfitting. השיטה יוצרת מספר מודלים עצמאיים, שכל אחד מהם מאומן על תת-קבוצה שונה של הנתונים. בסופו של דבר, תוצאות המודלים נשקללות יחד כדי לקבל תחזית סופית. המטרה היא לשפר את יציבות המודל ולמנוע התאמה יתרה לנתוני האימון, על ידי צמצום השונות והפחתת הסיכון של אוברפיטינג.
* Boosting - טכניקת Ensemble שמטרתה להפחית את ההטיה (bias) ולשפר את הדיוק של המודל. השיטה יוצרת סדרת מודלים, שכל אחד מהם מאומן על הנתונים כך שהוא מתמקד בתיקון טעויות המודל הקודם. כל מודל חדש מתמקד בדוגמאות שהמודלים הקודמים טעו בהן, ובסופו של דבר, התחזיות משולבות יחד עם משקל גבוה יותר למודלים המאוחרים יותר. המטרה היא לשפר את דיוק המודל על ידי תיקון טעויות המודלים הקודמים, אך יש לשים לב שהשיטה רגישות ל-overfitting אם הנתונים לא מספיק מגוונים או אם המודל מורכב מדי.

**אלגוריתם Random Forest:**

Random Forest הוא אלגוריתם המבוסס על Bagging (Bootstrap Aggregating), המשתמש במספר עצי החלטה כדי לשפר את הדיוק והיציבות של המודל. כל עץ מאומן על דגימה אקראית של הנתונים, והפיצול בכל עץ נעשה על תכונות אקראיות, מה שמפחית את הקשר בין העצים. תחזית הסופית מתקבלת מממוצע תחזיות העצים במקרה של רגרסיה (או מרוב קולות במקרה של סיווג).

היתרון המרכזי של ה-Random Forest הוא שמניעת Overfitting על ידי יצירת מודלים שונים על דגימות שונות של הנתונים. זה מספק תחזיות יציבות ומדויקות יותר בהשוואה לעץ החלטה בודד. ה-Random Forest גם מפחית את הרגישות לרעש בנתונים, משפר את היכולת להכליל על נתונים חדשים, ומצמצם טעויות על ידי מגוון חיזויים ממספר מודלים.

איך עובד ה-Random Forest?

* Bootstrap Aggregating (Bagging): כל עץ מקבל דגימה אקראית מהנתונים המקוריים (עם חזרה), כך שכל עץ לומד על תת-קבוצה שונה של הנתונים. זה מפחית את הסיכון ל-Overfitting ומגביר את המגוון בין העצים.
* Random Feature Selection: בכל פיצול בעץ, האלגוריתם בוחר תת-קבוצה אקראית של פיצ'רים ולא את כל הפיצ'רים. זה מונע עצים דומים מדי ומגדיל את המגוון של התחזיות.
* בניית עצים מרובים: כל עץ ביער (Forest) נבנה על תת-קבוצה שונה של הנתונים והפיצ'רים ומבצע חיזוי עצמאי. כל עץ יכול להחזיר תחזית שונה על אותם נתונים.
* חיזוי ממוצע: לאחר שכל העצים מבצעים את החיזוי שלהם, התוצאה הסופית היא ממוצע של תחזיות כל העצים. בשיטה זו, השפעת הטעויות של כל עץ נחלשת, ותחזית הסופית מדויקת ויציבה יותר.

1. תיאום פרמטרים (Hyperparameter Tuning) לRandom Forest

בחרנו להפעיל את טכניקת ה- GridSearchCV כדי למצוא את השילוב הטוב ביותר של פרמטרים עבור המודל. במהלך התהליך, GridSearchCV בודק מספר שילובים של פרמטרים:

* n\_estimators: אפשרויות שונות למספר העצים שנבדוק (50 ו-100).
* max\_depth: הערכים האפשריים עבור עומק העץ (10 ו-None).
* min\_samples\_split: הערכים האפשריים למספר הדוגמאות המינימלי שנדרשות לפיצול כל צומת (5 ו-10).
* cv=3: מספר החלקים ב-cross-validation. כאן הוגדר כ-3, כלומר התהליך יבצע חיתוך של הנתונים ל-3 חלקים ויבצע אימון על שני חלקים ובדיקה על החלק השלישי (כך נוודא שהמודל לא מותאם יתר על המידה).
* param\_grid\_rf: מדובר במילון שמכיל את האפשרויות השונות שנבדוק עבור כל פרמטר.
* n\_jobs=-1: פרמטר זה מציין שכאשר האלגוריתם רץ, ישתמש בכל הליבות של המעבד (אם יש יותר מליבה אחת), מה שמאיץ את התהליך של האימון.
* verbose=1: פרמטר זה קובע את רמת הפלט שהמודל ייתן במהלך הריצה. כאשר הערך הוא 1, המודל יציג עדכונים על התהליך של ה-GridSearch (כמו כמה שילובים נבדקים ועד כמה התהליך מתקדם). אם תציב את הערך ל-0, לא יופיעו עדכונים, ואם תציב ערך גבוה יותר (למשל, 2), יינתן יותר מידע על תהליך החישוב.

יש שיפור משמעותי בביצועים של המודל אחרי כיוונון הפרמטרים, עם ירידה בשגיאות (MSE ו-RMSE) ועם עלייה ביכולת להסביר את השונות במחירים (R²). ה-MAE מצביע על שיפור קל גם בהפחתת הטעויות המוחלטות.

1. Feature Importance עבור ה Random Forest

חשיבות המאפיינים בהתבסס על איך שהאלגוריתם עובד:

בשלב של Feature Importance בעצי החלטה, כולל Random Forest, נמדדת חשיבות המאפיינים לפי איך שהאלגוריתם מבצע את הפיצולים של הנתונים בעץ. כל צומת בעץ מתבצע על בסיס המאפיינים שמפחיתים את השונות בקבוצות שנוצרות. ככל שמאפיין תורם יותר לפיצול "נקי" ומפחית את השונות בצורה משמעותית, הוא ייחשב כחשוב יותר.

האלגוריתם קובע את מידת החשיבות של כל מאפיין לפי איך שהוא תורם לפיצול של הנתונים בעץ, כלומר עד כמה הוא מסייע בהפרדה בין קבוצות נתונים ומסייע בשיפור ביצועי המודל. חשיבות המאפיינים נמדדת לפי השפעתם על יכולת המודל לבצע החלטות נכונות ומדויקות.

חשיבות המאפיינים בהתייחסות לאלגוריתם כ"קופסה שחורה" (Black Box):

כשמתייחסים לאלגוריתם כ"קופסה שחורה", אנו לא בודקים כיצד המודל מבצע את ההחלטות או פועל מבפנים, אלא מתמקדים בהשפעת המאפיינים על ביצועי המודל בצורה נייטרלית. גישה זו מבוצעת באמצעות טכניקת Permutation Importance, שבה מערבבים את ערכי המאפיין ומודדים איך השיבוש הזה משפיע על תחזית המודל. בתהליך זה, השיבוש מערער את הקשר של המאפיין עם המשתנה התלוי, ולאחר מכן מודדים את ירידת ביצועי המודל. ירידה משמעותית בביצועים מעידה על כך שהמאפיין חשוב יותר לחיזוי, והמנגד, מאפיינים שגורמים לפחות ירידה נחשבים לפחות חשובים. היתרון של גישה זו היא שהיא אינה דורשת הבנה מעמיקה של המודל עצמו, והיא יכולה להתבצע על כל מודל, גם על מודלים מורכבים כמו Random Forest או Neural Networks. באמצעות טכניקה זו, ניתן לזהות אילו מאפיינים משפיעים ביותר על תחזית המודל מבלי להיכנס לפרטים על איך המודל פועל בפנים.

1. הפרדה לפי אתר- השוואת ביצועי המודל בין אתרים (Kayak ו-Momondo)

בשלב הזה, אנו משווים את ביצועי המודל אחרי שנאמן אותו על שני אתרים שונים: Kayak ו-Momondo. השתמשנו באותו המודל (Random Forest) אך חישבנו את מדדי הביצועים של כל אתר בנפרד.

המודל עבד טוב יותר על Kayak, אבל היו טעויות גדולות יותר על Momondo.

הביצועים השונים בין האתרים עשויים לנבוע מכך שהמאפיינים או הגורמים שמשפיעים על מחירי הטיסות ב-Kayak עשויים להיות שונים מאלו שב-Momondo. לדוגמה, ייתכן שגורמים כמו זמן יציאה או מרחקים בין נמלי תעופה משפיעים באופן שונה בין שני האתרים.