**הסבר על אלגוריתם XGBoost**

XGBoost (Extreme Gradient Boosting) הוא אלגוריתם למידת מכונה מבוסס עצים שמשתמש בשיטת **Gradient Boosting** לשיפור דיוק התחזיות. האלגוריתם בנוי כך שהוא ממזער שגיאות באופן הדרגתי על ידי הוספת עצים חדשים שמתמקדים בדוגמאות עם שגיאות גבוהות יותר.

**איך האלגוריתם עובד?**

1. **אתחול המודל** – האלגוריתם מתחיל עם תחזיות בסיסיות, כמו ממוצע התוויות.
2. **יצירת עצים ברצף** – כל עץ חדש לומד משגיאות העצים הקודמים, תוך התאמתו כדי לתקן את השגיאות.
3. **חישוב גרדיאנט** – האלגוריתם מחשב את השגיאות של התחזיות הקודמות כדי לקבוע כיצד לשפר את המודל.
4. **שקלול העצים** – כל עץ מקבל משקל בהתאם ליכולתו לתקן שגיאות של קודמיו.
5. **סכימת התחזיות** – התחזיות של כל העצים משוקללות יחד לקבלת התחזית הסופית.

**יתרונות של XGBoost**

✅ **מהירות ויעילות** – האלגוריתם מהיר מאוד בזכות פרלליות פנימית ואופטימיזציות של שימוש בזיכרון. ✅ **מניעת אוברפיטינג** – כולל מנגנונים כמו regularization (L1, L2) למניעת התאמה יתרה. ✅ **תמיכה בנתונים חסרים** – האלגוריתם מסוגל להתמודד עם ערכים חסרים ללא צורך בהשלמתם מראש. ✅ **גמישות** – מתאים לבעיות רגרסיה, סיווג ודירוג (Ranking).

**חסרונות של XGBoost**

❌ **רגישות לאוברפיטינג** – אם לא מכוונים היטב את הפרמטרים, המודל עלול ללמוד את רעשי האימון ולא להכליל היטב. ❌ **דרישות חישוביות גבוהות**– השימוש בהרבה עצים ופרמטרים מתוחכמים מצריך זמן חישוב וזיכרון גבוהים. ❌ **קשה להבנה**– לעומת מודלים פשוטים כמו רגרסיה לינארית, קשה להסביר למשתמשים כיצד המודל מקבל החלטות.

**הסבר על הפרמטרים שנבחרו**

במהלך האימון, השתמשנו ב-**GridSearchCV** כדי למצוא את הפרמטרים האופטימליים עבור האלגוריתם. להלן הסבר על כל פרמטר:

* **n\_estimators -** קובע את מספר העצים במודל. ערך גבוה מדי עלול לגרום לאוברפיטינג.
* **learning\_rate -** מגדיר בכמה כל עץ חדש תורם לתיקון השגיאה. ערך נמוך מדי יאריך את זמן האימון אך ימנע אוברפיטינג.
* **max\_depth -** שולט על מורכבות העצים. עצים עמוקים מדי עלולים לגרום לאוברפיטינג.
* **Subsample -** מגדיר את אחוז הדגימות מהנתונים שמשמש בכל סיבוב. ערך נמוך מסייע במניעת אוברפיטינג.
* **colsample\_bytree** - אחוז הפיצ'רים שנבחרים לכל עץ. ערך נמוך מסייע בהפחתת אוברפיטינג.
* **Gamma -** קובע מתי כדאי לבצע פיצול נוסף בעץ. ערך גבוה ימנע פיצולים מיותרים.
* **min\_child\_weight -** מונע יצירת צמתים שמכילים מעט מדי נתונים, ובכך מונע אוברפיטינג.

**מסקנה**

XGBoost הוא אחד האלגוריתמים החזקים ביותר ללמידת מכונה, עם יכולות גבוהות לניבוי מדויק ומהיר. עם זאת, יש להגדיר היטב את הפרמטרים כדי למנוע התאמת יתר לנתוני האימון. שימוש נכון ב- **GridSearchCV** ובקריטריונים להפחתת מורכבות המודל מאפשר להגיע לאיזון בין ביצועים טובים לבין יכולת הכללה טובה על נתונים חדשים.

פירוט מורחב:

**XGBoost (Extreme Gradient Boosting)**

XGBoost הוא אלגוריתם למידת מכונה מבוסס Gradient Boosting, המשפר תחזיות על ידי שילוב של עצים רבים.

זהו אלגוריתם אנסמבל (Ensemble Learning) מסוג Boosting, שבו כל עץ חדש מתקן את השגיאות של קודמיו, וכך המודל משתפר בהדרגה. במקום להסתמך על עץ יחיד, XGBoost משתמש בקבוצה של עצים הפועלים יחד, כאשר כל עץ תורם לשיפור התחזיות של המודל.

השוואה ל Random Forest:

XGBoost ו-Random Forest הם שניהם אלגוריתמי אנסמבל (Ensemble Learning), אך הם פועלים בגישות שונות.

XGBoost משתמש ב-Boosting, כלומר הוא בונה עצים באופן סדרתי, כך שכל עץ חדש מתקן את השגיאות של קודמיו, מה שהופך אותו לאלגוריתם מדויק במיוחד אך רגיש יותר לאוברפיטינג. לעומתו, Random Forest משתמש ב-Bagging, כלומר הוא בונה עצים במקביל, כאשר כל עץ מקבל מדגם אקראי של הנתונים ועובד באופן עצמאי, מה שמקטין את הסיכון לאוברפיטינג אך לעיתים פחות מדויק.

XGBoost לרוב מהיר יותר בזכות אופטימיזציות מתקדמות, בעוד Random Forest נחשב ליציב ומגוון יותר, במיוחד כאשר יש נתונים רועשים או חסרים.

איך האלגוריתם עובד?

1. אתחול – מתחילים עם תחזית בסיסית, למשל ממוצע הנתונים.
2. בניית עצים – כל עץ חדש מתקן את השגיאות של העץ הקודם.
3. חישוב גרדיאנט (Gradient) – האלגוריתם מחשב כמה רחוקה כל תחזית מהערך האמיתי.
4. שקלול עצים – לכל עץ משקל בהתאם ליכולתו לשפר את המודל.
5. סיכום התחזיות – התחזיות מכל העצים מתחברות לניבוי מדויק יותר.

**יתרונות של XGBoost:**

* מהירות ויעילות – האלגוריתם מהיר מאוד בזכות עיבוד מקבילי ואופטימיזציה של זיכרון.
* התמודדות עם אוברפיטינג – יש לו מנגנונים כמו Regularization (L1 ו-L2) שמונעים התאמה מוגזמת לנתוני האימון.
* תמיכה בערכים חסרים – האלגוריתם יודע להתמודד עם נתונים חסרים בלי צורך להשלימם מראש.
* גמישות גבוהה – אפשר להשתמש בו גם לרגרסיה, גם לסיווג וגם לדירוג (Ranking).
* בגרסאות החדשות של XGBoost (מ-2021 והלאה) קיימת תמיכה מובנית במשתנים קטגוריים, ללא צורך בהמרתם ל-One-Hot Encoding. במקום להשתמש ב-pd.get\_dummies(), ניתן פשוט להגדיר את העמודות כקטגוריות (category type בפנדס), והאלגוריתם יטפל בהן ישירות באמצעות הפרמטר enable\_categorical=True. היתרונות בשיטה זו הם חיסכון בזיכרון, האצת תהליך האימון על ידי מניעת "התפוצצות ממדית" (Curse of Dimensionality), וצמצום הצורך בהנדסת נתונים ידנית.

**חסרונות של XGBoost:**

* רגישות לאוברפיטינג – אם לא מכוונים את הפרמטרים נכון, המודל עלול לזכור את נתוני האימון יותר מדי.
* כבד חישובית – מודלים גדולים מצריכים משאבים רבים (מעבד וזיכרון).
* מורכבות גבוהה – קשה להסביר את התחזיות לעומת מודלים פשוטים כמו רגרסיה ליניארית.

**שלבי עבודה במחברת:**

אימון מודל ראשוני:

בשלב הראשון אימנו מודל ראשוני של XGBoost לרגרסיה, שמטרתו חיזוי מחירי טיסות והערכת הביצועים שלו.

הגדרות חשובות במודל:

* objective='reg:squarederror' – משמש לרגרסיה (ניבוי ערכים רציפים).
* enable\_categorical=True – מאפשר למודל לעבוד עם משתנים קטגוריים ללא המרה לדאמיז.
* random\_state=101 – מבטיח תוצאות עקביות בכל הרצה.

תוצאות המודל הראשוני היו מדויקות עם שגיאות נמוכות.

שלב שני – בחירת הפיצ'רים החשובים ושיפור המודל:

בשלב זה זיהינו את הפיצ'רים המשפיעים ביותר על מחיר הטיסה על מנת לשפר את המודל על ידי הסרת משתנים מיותרים כדי להפחית רעש ולמנוע Overfitting. לאחר הסרתם הרצנו את המודל מחדש עם הפיצ'רים הרלוונטיים בלבד כדי לבדוק את השפעת הסינון.

ציון ה-R² ירד מעט, כלומר המודל מסביר פחות מהשונות במחיר בהשוואה לגרסה הקודמת. ולמרות הירידה, השינויים קריטיים כדי להימנע מאוברפיטינג ולהבטיח שהמודל מתבסס על נתונים אמיתיים.

שלב שלישי – כוונון פרמטרים (Hyperparameter Tuning) ושיפור ביצועי המודל:

בשלב זה כווננו את הפרמטרים של מודל XGBoost כדי לשפר את ביצועיו, להפחית Overfitting ולשפר את יכולת ההכללה שלו על נתונים חדשים. לשם כך השתמשנו ב- GridSearchCV כדי לבדוק שילובים שונים של פרמטרים ולבחור את הערכים האופטימליים.

התאמנו את מספר העצים (n\_estimators), עומק העצים (max\_depth), קצב הלמידה (learning\_rate), גודל הדגימה (subsample), ופרמטרים נוספים למניעת Overfitting.

הפרמטרים החשובים ביותר ב-XGBoost:

* n\_estimators – מספר העצים במודל. ערך גבוה מדי עלול לגרום לאוברפיטינג.
* learning\_rate – מגדיר את קצב הלמידה של המודל. ערך נמוך מדי יאט את האימון, אך יסייע במניעת Overfitting.
* max\_depth – עומק מקסימלי של העצים. עצים עמוקים מדי ילמדו את הדאטה יותר מדי (Overfitting), בעוד עומק נמוך עלול לפספס דפוסים חשובים.
* subsample – אחוז הדוגמאות שמשתמשים בהן לכל עץ. ערך נמוך יותר מקטין Overfitting בכך שהוא מאלץ את המודל ללמוד ממדגמים שונים.
* colsample\_bytree – אחוז הפיצ'רים שמשתמשים בהם בכל עץ. עוזר למנוע תלות בין העצים ולשפר הכללה.
* gamma – קובע את רגולציית הפיצולים בעץ. ערך גבוה יותר ימנע פיצולים מיותרים, מה שיכול למנוע אוברפיטינג.
* min\_child\_weight – מונע יצירת צמתים קטנים מדי בעצים, וכך מסייע להקטין אוברפיטינג.

כוונון הפרמטרים שיפר משמעותית את דיוק המודל, הקטין את השגיאות והפך אותו ליציב ומדויק יותר לחיזוי מחירי טיסות.

שלב אחרון – ניתוח ביצועי המודל ובדיקת דיוק התחזיות:

בשלב זה בדקנו את דיוק התחזיות של המודל על ידי השוואת המחירים האמיתיים למחירים החזויים וניתוח השגיאות (Errors). חישבנו את הפערים בין הערכים כדי לבדוק עד כמה התחזיות קרובות למציאות, ובחנו האם המודל מעריך יתר על המידה ⬆ או פחות מדי ⬇. נמצא כי המודל מספק תחזיות מדויקות יחסית, עם קורלציה טובה בין מחיר אמיתי למחיר חזוי, והשאריות (Residuals) מפוזרות באופן אחיד, מה שמעיד על כך שהמודל אינו מוטה לכיוון מסוים. עם זאת, המודל מתקשה בעיקר בחיזוי טיסות יקרות מאוד, ככל הנראה עקב מחסור בדוגמאות בטווח מחירים זה. בהתבסס על ניתוח החשיבות של הפיצ'רים, ניתן לבחון הוספה או הסרה של משתנים מסוימים כדי לשפר את הדיוק.

**מסקנה סופית - XGBoost הוא המודל הטוב ביותר לניבוי מחירי טיסות**

לאחר השוואת ביצועי מספר אלגוריתמים לניבוי מחירי טיסות, XGBoost הוכיח את עצמו כמודל המדויק והיציב ביותר.

✔ XGBoost השיג את ציון ה-R² הגבוה ביותר (0.9037) – כלומר, הוא מסביר כ-90% מהשונות במחירי הטיסות, הרבה יותר מכל שאר המודלים שנבדקו.

✔ השגיאה הריבועית הממוצעת (RMSE) שלו היא הנמוכה ביותר (64.29) – מה שמעיד על כך שהתחזיות שלו קרובות מאוד למחירי הטיסות בפועל.

✔ XGBoost הפיק את ערך MSE (שגיאה ריבועית ממוצעת) הנמוך ביותר (4133.21) – כלומר, התחזיות שלו מדויקות יותר באופן כללי.

✔ השגיאה המוחלטת הממוצעת (MAE) הייתה הנמוכה ביותר (47.14) – מה שמראה שההפרש הממוצע בין המחירים האמיתיים לחזויים הוא הקטן ביותר.

XGBoost הוכיח את עצמו כמודל הטוב ביותר לחיזוי מחירי טיסות בהשוואה לשאר האלגוריתמים שנבדקו. בהשוואה ל-Random Forest, הוא סיפק דיוק גבוה יותר ושגיאות נמוכות יותר, תוך שימוש באופטימיזציות שהפכו אותו למהיר ויעיל יותר. בהשוואה ל-Decision Tree, שמושפע מאוד משינויים בנתונים, XGBoost משתמש בגישת Boosting שמאפשרת לו ללמוד ולהשתפר בכל שלב, מה שהופך אותו ליציב ומדויק יותר. בהשוואה ל-K Nearest Neighbors, שהתמודד בקושי עם דאטה מורכב כמו מחירי טיסות, XGBoost הצליח לזהות דפוסים משמעותיים בצורה טובה יותר ולספק תחזיות מדויקות יותר. בהשוואה ל-Support Vector Regression (SVR), שנכשל בלמידת הנתונים והציג שגיאות גבוהות מאוד, XGBoost הצליח להתמודד עם הדאטה הרב-ממדי ולספק תוצאות טובות בהרבה.