**אלגוריתמים שבדקנו לחיזוי מחיר הטיסה**

**רגרסיה לינארית - Linear Regression:**

רגרסיה לינארית הוא מודל ראשוני שהשתמשנו בו שהצביע על כך שהקשר בין המחירים של הטיסות למשתנים השונים הוא לא לינארי. למרות שאימנו את המודל על מגוון משתנים והשתמשנו בטכניקות שונות לשיפור הביצועים, הגענו לתוצאה שמצביעה על כך שהמודל אינו מצליח לתפוס את המורכבות של מחירי הטיסות. ה-R² הנמוך שהתקבל יחד עם המדדים האחרים שמצביעים על דפוסים לא אקראיים, משמעות הדבר שהרגרסיה הלינארית אינה מתאימה למערכת היחסים הזו, ולכן יש צורך במודלים לא-לינאריים שיכולים להתמודד עם קשרים מורכבים יותר כדי לשפר את דיוק החיזוי.

|  |  |  |  |
| --- | --- | --- | --- |
| R2 | MAE | MSE | RMSE |
| 0.408 | 124.23 | 25322.93 | 159.13 |

**Support Vector Regressor :**

בשימוש באלגוריתם SVR נתקלנו בכמה בעיות. תחילה, ניסינו את גרעין ה-RBF, שמתאים לזיהוי קשרים לא ליניאריים, אך סיבוכיות החישוב הגבוהה של המודל הפכה אותו ללא פרקטי לדאטה גדול. לאחר מכן, ניסינו לחלק את הדאטה למקטעים, אך גם זה לא שיפר את הביצועים. עברנו למודל LinearSVR, שמתאים יותר לדאטה גדול, אך גם הוא לא הצליח להתמודד עם הקשרים הלא ליניאריים במחירי הטיסות. כל הניסיונות לשפר את המודל באמצעות כוונון פרמטרים לא הובילו לשיפור משמעותי. המסקנה היא ש-SVR, במיוחד עם קרנל ליניארי, לא מתאים לכמות רבה של דאטה ולמשימה שלנו – חיזוי מחירי טיסות שמערבים קשרים מורכבים ולא ליניאריים.

|  |  |  |  |
| --- | --- | --- | --- |
| R2 | MAE | MSE | RMSE |
| 0.2636 | 129.03 | 31605.17 | 177.78 |

**Decision Tree Regressor:**

עץ החלטה לרגרסיה הוא אלגוריתם שמספק חיזוי של ערכים רציפים, במקרה שלנו – חיזוי מחירי טיסות. המודל בונה עץ שמחלק את הנתונים בצמתים כך שהשונות בתוך כל קבוצה תהיה מינימלית. כל עלה בעץ מהווה תחזית שהיא ממוצע הערכים בקבוצה המתאימה. היתרון של עץ החלטה הוא בכך שהוא מאפשר חיזוי ללא צורך בהנחות על הקשרים הלינאריים בין המשתנים, ומתמודד היטב עם נתונים שיש בהם קשרים לא לינאריים. יחד עם זאת, אחד החסרונות הוא הנטייה ל-Overfitting כאשר העץ עמוק מדי, דבר שמוביל לחיזוי מדויק מדי על נתוני האימון, אך לא בהכרח על נתונים חדשים. כדי לשפר את ביצועי המודל, ביצענו ניתוח חשיבות מאפיינים, הסרנו מאפיינים פחות חשובים, ויישמנו כוונון פרמטרים באמצעות GridSearchCV כדי למצוא את הערכים האופטימליים. בסופו של דבר, המודל הצליח לבצע חיזויים טובים יחסית למחירים האמיתיים, אך עדיין נתקל בקשיים בחיזוי מחירים גבוהים, כפי שניתן לראות מהשגיאות הגדולות יותר בטיסות יקרות.

|  |  |  |  |
| --- | --- | --- | --- |
| R2 | MAE | MSE | RMSE |
| 0.8431 | 58.73 | 6735.38 | 82.07 |

**Random Forest Regressor:**

בהמשך לניסיון עם עץ החלטה, שדרגנו את המודל ל- Random Forest.

אלגוריתם המבוסס על שיטת Ensemble - משתמש בכמה עצים שמאומנים על דגימות אקראיות של הנתונים ובחירה אקראית של תכונות בכל פיצול. כל עץ מבצע חיזוי עצמאי, והתחזית הסופית מתקבלת על ידי חישוב ממוצע תחזיות כל העצים. טכניקת ה - Bagging שבה משתמש מאפשרת להפחית לו את ה-overfitting על ידי יצירת מודלים שונים שמפחיתים את השפעת טעויות בודדות, ומגוון העצים מבטיח תחזיות מדויקות ויציבות יותר. שיטה זו משפרת את דיוק המודל ומפחיתה את הרגישות לרעש בנתונים.

בנוסף, עבור Random Forest השתמשנו בשתי טכניקות לניתוח חשיבות התכונות (לצורך ביצועי אופטימיזציה ע"י התמקדות בתכונות החשובות):

* Feature Importance – שבה אנחנו מודדים את התרומה של כל תכונה ללמידת המודל ולביצועים שלו. טכניקה זו מאפשרת לנו להבין אילו תכונות משפיעות יותר על תחזית המודל.
* Permutation Importance – בשיטה זו, אנחנו מערבבים את הערכים של כל תכונה באופן רנדומלי ומודדים את השפעתה על ביצועי המודל. אם ביצועי המודל יורדים משמעותית לאחר הערבוב, זה מצביע על כך שהתכונה חשובה ותרומתה משמעותית לחיזוי המודל. כשמתייחסים למודל כ- "קופסא שחורה", המטרה היא להעריך את החשיבות של כל תכונה מבלי לדעת בדיוק איך כל עץ במודל מקבל את החלטותיו. הערבוב מאפשר להבין את השפעת כל תכונה על הביצועים הכוללים של המודל, מבלי להיכנס לפירוט של איך המודל מבצע את החיזוי בכל עץ בנפרד.

בנוסף, השתמשנו ב- Grid Search כדי לבצע חיפוש אוטומטי על הפרמטרים של המודל כמו מספר העצים, העומק המקסימלי של העצים, והמספר המינימלי של דגימות בכל פיצול. באמצעות Grid Search, הצלחנו למצוא את ההגדרות האופטימליות למודל, מה שעזר לנו לשפר את ביצועי המודל ולהימנע מ-overfitting.

|  |  |  |  |
| --- | --- | --- | --- |
| R2 | MAE | MSE | RMSE |
| 0.8495 | 57.85 | 6460.3 | 80.38 |

**K Nearest Neighbors :**

מדובר באלגוריתם פשוט אך גמיש שמתאים לרגרסיה. התחלנו בהרצת המודל עם הגדרות ברירת מחדל ולאחר מכן עברנו לכוונון פרמטרים באמצעות GridSearchCV כדי למצוא את הערכים האופטימליים. כווננו את K, מדד המרחק (בחרנו במרחק מנהטן), והשימוש במשקלים (בחרנו במשקל מבוסס מרחק), שהשפיעו מאוד על ביצועי המודל.

ב-KNN, התוצאה הסופית של התחזית מתקבלת על ידי חישוב ממוצע של הערכים של ה-K שכנים הקרובים ביותר לנקודת הקלט, כאשר המרחק בין נקודת הקלט לשכנים משפיע על הממוצע. אם K קטן מדי, המודל יהיה רגיש מאוד לרעש, ובמקרה של K גבוה מדי, המודל עלול לא להתאים בצורה אופטימלית לנתונים. בחירה ב-K=15, עם מרחק מנהטן ומשקל מבוסס מרחק, סיפקה איזון טוב בין רגישות לרעש והחלקת הנתונים, מה שעוזר להימנע מ-Overfitting ו-Underfitting.

הדיוק של המודל, כפי שמצוין על ידי R² גבוה (מעל 0.84), מצביע על כך שהמודל מצליח להסביר את השונות במחירי הטיסות בצורה טובה. עם זאת, יש מקום לשיפור בנוגע לטעויות בתחזיות של מחירים גבוהים, כפי שמצביעים ה-RMSE וה-MAE. בסך הכל, התוצאה נחשבת לטובה, במיוחד בהתחשב בכך שמדובר במודל פשוט יחסית.

|  |  |  |  |
| --- | --- | --- | --- |
| R2 | MAE | MSE | RMSE |
| 0.8458 | 57.6242 | 6619.582 | 81.3608 |

**Kernel Approximation:**

לא השתמשנו ב-Gaussian Process Regression (GPR) ישירות מכיוון שהאלגוריתם, למרות שהוא חזק וגמיש, מצריך חישוב של מטריצת קו-וריאנס שמורכבות החישוב שלה היא O(n³). זה הופך את השיטה ללא ישימה עבור מאגרי נתונים גדולים, כמו במקרה שלנו עם מעל 500,000 דגימות. ככל שמספר הדגימות גדל, הזמן והמשאבים הנדרשים עבור חישוב מטריצת הקו-וריאנס הולכים ומתרבים, דבר שמקשה מאוד על השימוש ב-GPR ישירות. כדי להתגבר על בעיה זו, השתמשנו ב-Nystroem Kernel Approximation שמפחית את העלות החישובים באמצעות ביצוע קירוב לפונקציית הקרנל באמצעות דגימה של תת-קבוצה של הנקודות.

בהתבסס על Gaussian Process Regression (GPR) בשילוב עם Ridge Regression ו- Nystroem Kernel Approximation, הצלחנו לבצע חיזוי מדויק של מחירי טיסות תוך התמודדות עם כמות הנתונים הגדולה ודרישות החישוביות הגבוהות של GPR.

הקרנל Laplacian נמצא כמתאים ביותר, ושיפר את הדיוק, במיוחד עבור נתונים לא לינאריים. הרגולריזציה של Ridge Regression (α=0.05) שיפרה את יציבות המודל והפחיתה את הסיכון ל-Overfitting.

תהליך כוונון היפר-פרמטרים באמצעות GridSearchCV כלל את בדיקת קרנלים שונים, מספר רכיבים ב-Nystroem, ודרגות רגולריזציה. בסופו של תהליך, עם n\_components=7000, הצלחנו להניב תוצאות טובות מאוד, עם R² של 0.8555, שיפור משמעותי בביצועים וירידה בשגיאות.

|  |  |  |  |
| --- | --- | --- | --- |
| R2 | MAE | MSE | RMSE |
| 0.8555 | 59.21 | 6202.68 | 78.76 |

**XGBOOST - התוצאות הטובות ביותר:**

האלגוריתם הטוב ביותר עבור המשימה הוא XGBoost (Extreme Gradient Boosting) מבין כל האלגוריתמים שנבדקו.

XGBoost, המבוסס על שיטת Boosting, בונה סדרה של עצים שבהם כל עץ חדש מתקן את השגיאות של קודמיו, מה שמוביל לשיפור הדרגתי של התחזיות. במהלך התהליך, כווננו את הפרמטרים של המודל באמצעות GridSearchCV, כולל בחירת n\_estimators, learning\_rate, max\_depth, ו-subsample. הכוונון הסופי הביא לתוצאות מצוינות בהן המודל מסביר כ-90% מהשונות במחירי הטיסות ובעל שגיאות נמוכות.

בנוסף, השתמשנו בשתי טכניקות לניתוח חשיבות התכונות לשיפור ביצועי המודל:

* Feature Importance – למדנו את תרומת כל תכונה למודל ולביצועיו.
* Permutation Importance – באמצעות ערבוב אקראי של תכונות, למדנו אילו תכונות משפיעות ביותר על תחזיות המודל.

בהשוואה לשאר המודלים, XGBoost היה המודל המדויק ביותר ונתן תחזיות עם שגיאות נמוכות יותר, מה שהופך אותו לאלגוריתם המוצלח ביותר עבור משימה זו.

|  |  |  |  |
| --- | --- | --- | --- |
| R2 | MAE | MSE | RMSE |
| **0.9037** | **47.14** | **4133.21** | **64.29** |