**Gaussian Process Regression – תהליך גאוסיאני לרגרסיה:**

Gaussian Process (GP) - תהליך גאוסי הוא מודל הסתברותי שמתאר התפלגות של פונקציות רציפות, כמו לאורך ציר הזמן או במרחב, בהנחה שכל אוסף סופי של משתנים מקריים מתפלג נורמלית רב-ממדית.

Gaussian Process Regression (GPR) משמש בעיקר לבעיות רגרסיה, כלומר חיזוי ערכים רציפים, ומציע יתרונות ייחודיים כמו אומדן אי-ודאות וגמישות במידול נתונים מורכבים. האלגוריתם נפוץ בתחומים כמו חיזוי סדרות זמן, מחירים וטמפרטורה, אופטימיזציה של פונקציות לא ידועות (Black Box Optimization) והערכת רמת ביטחון בתחזיות, מה שהופך אותו לכלי חיוני בניתוח רפואי, זיהוי הונאות ויישומים נוספים.

מאפיינים מרכזיים של האלגוריתם:

* ייצוג פונקציות כמשתנים מקריים: במקום לקבוע נוסחה קבועה לרגרסיה, GPR מגדיר התפלגות על פונקציות, המאפשרת חיזוי נקודות חדשות עם רמת אי-ודאות.
* חיזוי לאורך ציר זמן: האלגוריתם מתאים במיוחד לבעיות ניבוי סדרות זמן, שכן הוא ממזער את השונות בין נקודות מדידה ומשפר את הדיוק של התחזיות.
* התאמה דינמית: המודל אינו פרמטרי ואינו מוגבל לצורה מתמטית קבועה מראש, אלא מתאים את עצמו לנתונים הקיימים.

רכיבים מרכזיים בהתפלגות הגאוסיאנית

בתהליך גאוסיאני, החיזוי מבוסס על שלושה רכיבים מרכזיים המגדירים את מבנה ההתפלגות ומאפשרים חיזוי מדויק תוך מדידת אי-ודאות.

* וקטור ממוצעים (μ): מייצג את הערכים הצפויים עבור כל נקודת נתונים. משמש כבסיס לחיזוי.
* סטיית תקן (σ): משקפת את רמת אי-הוודאות סביב הממוצע. ככל שסטיית התקן גדולה יותר, כך רמת הביטחון בתחזית נמוכה יותר.
* מטריצת קו-וריאציה (Σ): מתארת את התלות בין נקודות הנתונים. ככל שהקו-וריאנס בין נקודות גבוה יותר, כך החיזוי מדויק יותר.
* מרכיב נוסף הוא הקרנל (Kernel)

**פונקציות גרעין (Kernel Functions) בתהליך גאוסיאני:**

הקרנל (Kernel) הוא רכיב מרכזי בתהליך הגאוסיאני, שכן הוא מכתיב כיצד מחושבת מטריצת הקו-וריאנס (Σ) וקובע את הקשרים בין נקודות שונות במרחב הנתונים. תפקידו למדוד דמיון בין נקודות – ככל שהן קרובות יותר, כך הקשר ביניהן חזק יותר. בכך, הקרנל משפיע ישירות על מבנה מטריצת הקו-וריאנס, מה שקובע את איכות התחזיות והיכולת של המודל ללמוד דפוסים מתוך הנתונים

תפקיד הקרנל:

* קובע את התלות בין נקודות נתונים על ידי מדידת מידת הדמיון ביניהן.
* מאפשר למודל להתמודד עם מבנים לא לינאריים על ידי שינוי ייצוג הנתונים.
* משפיע על גמישות המודל ודיוק התחזיות בהתאם לסוג הקרנל שנבחר.

קרנלים נפוצים:

* RBF (Radial Basis Function): מתאים לקשרים חלקים ורציפים, הקרנל הנפוץ ביותר.
* Linear Kernel: מתאים למערכות לינאריות.
* Polynomial Kernel: מתאים לקשרים פולינומיאליים.
* Matern Kernel: מתאים לנתונים רועשים ומחוספסים.
* White Kernel: משמש לייצוג רעש בנתונים

אתגרים בעבודה עם קרנלים:

* מורכבות גבוהה – קשה לפרש אינטואיטיבית קרנלים מורכבים.
* בחירה לא טריוויאלית – לרוב מתבצעת בניסוי וטעייה.
* עלות חישובית גבוהה – מודלים גאוסיאניים דורשים משאבים חישוביים משמעותיים, במיוחד עם דאטה גדול.

איך האלגוריתם עובד?

1. הנחת קשרים בין נקודות - האלגוריתם מניח שיש קשר סטטיסטי בין כל הנקודות במרחב. קשר זה נקבע על ידי פונקציית הקרנל (Kernel), שמודדת את מידת הדמיון בין נקודות שונות. ככל ששתי נקודות קרובות יותר במרחב, כך הקשר ביניהן חזק יותר.
2. חיזוי ערכים חדשים - בהינתן נקודות עם ערכים ידועים, האלגוריתם משתמש בקרנל כדי להעריך את מידת הדמיון שלהן לנקודה חדשה. על בסיס קשרים אלו, ניתן לבצע חיזוי של הערך החדש בצורה רציפה ומדויקת, תוך שקלול אי-הוודאות של התחזית.
3. מטריצת הקו-וריאנס - מטריצת Covariance Matrix מתארת את התלות בין הנקודות בנתונים. קשרים אלה מאפשרים למודל להשתמש במידע על נקודות קיימות כדי לאמוד ערכים של נקודות חדשות. ככל שהקו-וריאנס בין נקודות גבוה יותר, כך החיזוי יהיה אמין יותר.

למידה אקטיבית

למידה אקטיבית היא שיטה חיצונית לשיפור מודלים כמו Gaussian Process Regression (GPR), אך אינה חלק מובנה מהאלגוריתם עצמו. בתהליך זה, המודל מתחיל עם כמות קטנה של נתוני אימון (Train Set) כדי לחסוך במשאבים חישוביים ולבצע חיזוי ראשוני, תוך שימוש בסט בדיקה (Test Set) גדול יחסית להערכת ביצועיו. אם המודל שוגה בתחזיותיו, מתבצע עדכון דינמי, שבו מוסיפים נתונים חדשים לסט האימון כך שהמודל מתאים את עצמו לנקודות חדשות ולומד מהטעויות. שיטה זו חוסכת משאבים, משפרת את הדיוק בהדרגה, ומותאמת במיוחד למצבים שבהם איסוף נתונים הוא יקר או מוגבל (כגון בדיקות רפואיות). עם זאת, GPR אינו מבצע למידה אקטיבית כברירת מחדל, וניתן להשתמש בו גם ללא תהליך זה.

יתרונות וחסרונות של Gaussian Process Regression (GPR)

✅ יתרונות:

✔ מדידת אי-ודאות: האלגוריתם לא רק מספק חיזוי אלא גם טווח ביטחון, מה שמאפשר להעריך את רמת הביטחון בתחזיות.

✔ גמישות גבוהה: יכול להתאים כמעט לכל פונקציה באמצעות Kernels שונים, מה שמאפשר עבודה עם מבנים נתונים מורכבים.

✔ התאמה אישית עם Kernels: ניתן לבחור או לשלב Kernels שונים כדי לשפר את הדיוק ולהתאים את המודל לבעיה.

✔ למידה אקטיבית: המודל מתעדכן ומשתפר ככל שנוספים לו נתונים חדשים, תכונה שימושית באופטימיזציה בייזיאנית.

✔ חיזוי בנתונים חסרים: האלגוריתם יכול לבצע חיזוי גם כאשר חסרים נתונים מסוימים, על בסיס תלות בין הנקודות.

✔ אומדן אי-ודאות: מאפשר זיהוי תחומים שבהם המודל פחות בטוח, שימושי בתחומים קריטיים כמו רפואה ופיננסים.

❌ חסרונות:

⚠ זמן חישוב ארוך: בעל מורכבות חישובית O(n³), ולכן אינו יעיל עבור מערכי נתונים גדולים במיוחד.

⚠ קושי בפרשנות: קשה להסביר את הקשרים בין הנתונים בהשוואה למודלים פרמטריים פשוטים, והצלחתו תלויה בבחירת Kernel נכונה.

⚠ Overfitting: ללא בחירה נכונה של Kernel והגבלות מתאימות, המודל עלול להתאים את עצמו לרעש בנתונים ולהוביל לתחזיות שגויות.

⚠ בעייתיות בנתונים עם שינויים חדים: GPR מתקשה להתמודד עם שינויים פתאומיים, מאחר שהוא מתבסס על פונקציות חלקות כמו RBF.

⚠ מגבלות חישוביות בנתונים מרובי ממדים: ככל שמספר המדדים (Features) גדל, חישוב מטריצת הקו-וריאנס הופך למסובך יותר, מה שמקשה על עבודה עם דאטה רחב מאוד.

📌 פתרונות אפשריים לבעיות חישוביות:

✔ סינון מאפיינים לא רלוונטיים כדי לצמצם את מורכבות הנתונים.

✔ שימוש ב-Kernels פשוטים יותר כדי להקטין את עומס החישוב.

✔ בחירה נכונה של length-scale כדי לאזן בין גמישות המודל לבין מניעת Overfitting.

**מה כדאי להוסיף להסבר המקדים?**

**🔹 למה לא השתמשנו ישירות ב-GPR?**

✔ **סיבוכיות חישובית גבוהה של GPR** → לא ניתן להריץ אותו על דאטה גדול (O(n³)).  
✔ **במקום זאת השתמשנו בקירוב קרנלים (Nystroem Kernel Approximation)**, שמאפשר להפוך את החישוב לאפשרי על דאטה גדול מבלי לוותר על היתרונות של שימוש בקרנלים.

**🔹 מהו קירוב קרנלים (Kernel Approximation) ולמה הוא חשוב?**

✔ **Nystroem Kernel Approximation** מאפשר להקטין את גודל מטריצת הקרנל ולשמור רק חלק קטן ממנה (למשל n\_components=7000 במקום חישוב מלא).  
✔ **במקום להשתמש ב-GPR ישירות, השתמשנו במודל רגרסיה (Ridge Regression) על הפיצ'רים שעברו את קירוב הקרנל.**

**🔹 איך זה מתקשר לקוד?**

✔ **במקום להשתמש ישירות ב-GPR**, אנחנו:  
1️⃣ **מבצעים קירוב קרנלים עם Nystroem(kernel='laplacian', n\_components=7000)** כדי להמיר את הנתונים למרחב לא-לינארי, כמו ש-GPR עושה.  
2️⃣ **משתמשים ברגרסיית Ridge עם alpha=0.05** כדי למנוע Overfitting ולמצוא קשרים בתוך המרחב החדש.

שלבי עבודה במחברת:

שלב ראשון:

1️⃣ שימוש ברגרסיה גאוסיאנית (Using Gaussian Regression)

סיבוכיות חישובית:

רגרסיה גאוסיאנית דורשת זיכרון חישובי של O(n3)O(n^3)O(n3) ולכן חשוב לבחור היטב את התכונות (Features) שמשפיעות על מחיר הטיסה. שימוש ביותר מדי נתונים יוביל לעומס חישובי גדול מאוד.

פתרונות אפשריים:

✔ הפחתת מספר השורות – במקום לעבוד על כל הנתונים, ניתן להקטין את מסד הנתונים ולבצע אימון על חלק קטן ממנו.

✔ שימוש ב-Feature Importance – מאחר שכבר חושבה חשיבות המשתנים (Feature Importance) באמצעות XGBoost, נבחר רק את 5 המשתנים החשובים ביותר ולא את כל המשתנים הזמינים.

התמודדות עם משתנים קטגוריים:

✔ GPR לא תומך בעמודות קטגוריות, ולכן יש להמיר אותן למשתנים דמי (Dummy Variables / One-Hot Encoding).

✔ הימנעות מהוספת משתנים רבים (כמו Airlines) – כי One-Hot Encoding מגדיל משמעותית את מספר המימדים, מה שיקשה על המודל.

שינוי האלגוריתם ל-Kernel Approximation עם Ridge Regression

למה שינינו את האלגוריתם?

✔ GPR עם 5000 דגימות בלבד נותן תוצאות R² נמוכות מאוד, גם כאשר השתמשנו רק ב-2 משתנים.

✔ לכן, במקום להשתמש ב-GPR ישירות, ננסה Kernel Approximation + Ridge Regression – שיטה שמדמה שימוש בקרנלים אך עם חישוב יעיל יותר.

מה נעשה עכשיו?

✔ נשתמש באותם משתנים שנבחרו ב-XGBoost, כי הם כבר הוכחו כחשובים.

✔ נוסיף תכונות חדשות כמו Return Weekday, TTT, Return Day כדי לשפר את המודל.

שלב שני:

הסבר: למה להשתמש בקירוב ניסטורם (Nystroem Approximation) במקום בתהליך גאוסיאני רגיל?

🔹 הבעיה עם Gaussian Process Regression (GPR):

רגרסיה גאוסיאנית רגילה היא גמישה ולא-פרמטרית, מה שהופך אותה למתאימה למודלים מורכבים.

עם זאת, החיסרון המרכזי של GPR הוא סיבוכיות חישובית של

𝑂

(

𝑛

3

)

O(n

3

) בגלל הצורך להפוך את מטריצת הקרנל.

במקרה של מערך נתונים גדול (500,000 טיסות), החישובים הופכים בלתי מעשיים עקב דרישות זיכרון וזמן חישוב.

🔹 הפתרון: Nystroem Approximation

Nystroem Approximation היא שיטה לקירוב פונקציית הקרנל של GPR באמצעות תת-מדגם מהנתונים.

היא מפחיתה את הסיבוכיות החישובית ל-

O(nm

2

), כאשר m הוא מספר המרכיבים שנבחרו (כלומר, גודל תת-המדגם).

כך ניתן לשמר את היתרונות של GPR, אך להפעיל אותו על דאטה רחב-היקף בצורה יעילה יותר.

🔹 שילוב עם Ridge Regression

לאחר שהקרנל מקורב באמצעות Nystroem, ניתן להשתמש ב-Ridge Regression כדי לבצע רגרסיה מהירה ויציבה.

השילוב הזה מאפשר לאזן בין דיוק החיזוי לבין חיסכון חישובי, ולכן הוא מתאים במיוחד לניתוח מחירי טיסות בקנה מידה גדול.

💡 מסקנה

במקום להשתמש ישירות ב-GPR, שבו העלות החישובית גבוהה מדי, אנו משתמשים בקירוב ניסטורם (Nystroem Approximation), המאפשר לנו לשמור על רגרסיה מבוססת קרנלים אך עם חישובים מהירים יותר. השילוב עם Ridge Regression נותן פתרון מאוזן בין דיוק לביצועים חישוביים.

שלב שלוש

**📌 שלב 2: נרמול (Normalization) של הנתונים**

מאחר שחלק מהתכונות הן **מספריות רציפות (כמו משך טיסה, יום יציאה וכו’)** וחלקן **קטגוריות (כמו חברות תעופה שהומרו ל-One-Hot Encoding)**, אנחנו מבצעים נרמול רק על **התכונות המספריות** כדי למנוע השפעה לא מאוזנת על המודל.

StandardScaler() מבצע נרמול סטטיסטי (Z-score normalization): כל משתנה מספרי יוקטן לטווח של ממוצע 0 וסטיית תקן 1.

חשוב בניתוחים מבוססי קרנלים (Kernel-based methods) כמו GPR, שכן סקיילינג משפיע על פעולת הקרנלים.

שלב ארבע

**📌 מטרת הקוד: שילוב קירוב קרנל ניסטורם (Nystroem) עם רגרסיית Ridge**

המודל כאן **משתמש בקירוב קרנל (Kernel Approximation)** על מנת **להאיץ את החישובים של GPR** ולהפוך את האלגוריתם לאפשרי עבור סט נתונים גדול.

שימוש ב-Nystroem Approximation לקירוב קרנל **Nystroem** היא טכניקה שמבצעת **קירוב של פונקציות קרנליות** באמצעות **תת-מדגם מהנתונים**, מה שמקטין את המורכבות החישובית

רגרסיית Ridge כמודל הלמידה

 **רגרסיית Ridge היא וריאציה של רגרסיה לינארית עם רגולריזציה (L2), שמונעת Overfitting.**

 מאחר שקרנלים עלולים לגרום למודל להיות רגיש לנתונים, Ridge מסייע למנוע התאמה יתרה.

✔ **משתמשים ב-Pipeline** – המודל כולל **שני שלבים:**

1. **Nystroem** – מבצע קירוב קרנל לנתונים.
2. **Ridge Regression** – משתמש בנתונים המומרמים כדי לבצע רגרסיה.

שלב חמש

✔ המודל **מתאמן** על סט האימון X\_train\_scaled, כלומר:

* תחילה **Nystroem מייצר את הייצוג המקורב של הקרנל**.
* לאחר מכן **רגרסיית Ridge מתאמנת על הנתונים** המבוססים על הקרנל.
* **מאחר שהמודל הוכשר על נתונים מנורמלים, גם התחזיות (y\_pred\_scaled) יהיו בסקייל של StandardScaler.**
* מאחר שהמודל אומן על נתונים מנורמלים, יש צורך להחזיר את התחזיות לטווח המחירים האמיתי (NIS).
* זה מתבצע באמצעות inverse\_transform() של ה-Scaler ששימש לנרמול y\_train\_scaled.

באופן כללי, המודל מבצע חיזויים סבירים, אך קיימים מרווחי שיפור.

שלב שש:

בשלב זה, אנו מבצעים כוונון היפרפרמטרים (Hyperparameter Tuning) באמצעות GridSearchCV, כדי למצוא את השילוב האופטימלי של פרמטרי Nystroem Kernel Approximation ו-Ridge Regression. נבדוק קרנלים שונים (rbf, poly, sigmoid, laplacian), מספר רכיבים (n\_components), דרגות פולינום (degree) ורמות רגולריזציה (alpha). לאחר מציאת הפרמטרים האופטימליים, נכשיר מחדש את המודל עם הנתונים המלאים, נבצע חיזוי, נשיב את המחירים לטווח המקורי (inverse\_transform) ונעריך את ביצועי המודל באמצעות מדדים כמו MSE, RMSE, R² ו-MAE. במידת הצורך, נבצע Fine-Tuning נוסף לשיפור הדיוק.

בשלב זה, כוונון ההיפרפרמטרים (GridSearchCV) הושלם בהצלחה, ונבחרו הפרמטרים האופטימליים: קרנל Laplacian, דרגה 3, n\_components=500, ו-ridge\_\_alpha=0.01. שיפורים נצפו בביצועי המודל, עם R² של 0.7539, MSE של 10,560.72, ו-RMSE של 102.77, המעידים על דיוק גבוה יותר וצמצום השגיאות. מאחר שהרצת GridSearchCV דורשת זמן רב, נשתמש ישירות בפרמטרים שנבחרו להמשך העבודה.

שלב 7:

בשלב זה, הגדלת **n\_components ל-7000** שיפרה משמעותית את ביצועי המודל, עם **עלייה ב-R² ל-0.8542** וירידה ניכרת במדדי השגיאה (**MSE, RMSE, MAE**). מאחר שהמחשב אינו יכול להתמודד עם n\_components=10000, זהו המקסימום שנוכל לבדוק. השלב הבא כולל **כוונון נוסף של ridge\_\_alpha** כדי למצוא את ערך הרגולריזציה האופטימלי, **בדיקת יציבות המודל עם קרוס-ולידציה (Cross-Validation)** כדי לוודא שהביצועים נשמרים גם בסטים שונים של נתונים, ולבסוף **שמירת המודל (Model Saving)** כך שנוכל להשתמש בו בעתיד ללא צורך באימון מחדש.

שלב 8:

בשלב זה, נבדקו ערכים שונים של ridge\_\_alpha (0.05, 0.1, 0.5, 1.0) כדי למצוא את רמת הרגולריזציה האופטימלית למודל. עבור כל ערך, המודל אומן מחדש ונבדק על סט הבדיקה, תוך מדידה של R², MSE, RMSE ו-MAE. המטרה הייתה לזהות את הערך המאזן בצורה הטובה ביותר בין גמישות המודל למניעת Overfitting או Underfitting. התוצאה הסופית מאפשרת לקבוע את ערך alpha שממזער את השגיאות ומגדיל את יכולת ההסבר של המודל (R² גבוה יותר).

בשלב זה, בוצע **כוונון נוסף של ridge\_\_alpha**, ונמצא כי **α = 0.05** מספק את התוצאות הטובות ביותר, עם **R² של 0.8555**, **MSE נמוך של 6202.68**, **RMSE של 78.76**, ו-**MAE של 59.21**, מה שמעיד על חיזוי מדויק יותר. לאחר בחירת הפרמטרים האופטימליים (nystroem\_\_kernel='laplacian', n\_components=7000, ridge\_\_alpha=0.05), המודל אומן מחדש ונבדק שוב, והתוצאות נותחו באמצעות השוואה בין מחירים בפועל למחירים חזויים. תוצאות אלו מוצגות בטבלה ובגרפים המציגים את **התפלגות השגיאות ודיוק החיזוי**, תוך התמקדות בהערכת כיוון הטעות (Over/Under Prediction).

**📌 סיכום כל שלבי התהליך - חיזוי מחירי טיסות באמצעות קירוב קרנל ורגרסיית Ridge**

**🔹 שלב 1: בחירת גישה אלגוריתמית**

מאחר שנתוני הטיסות כללו **500,000+ טיסות**, **Gaussian Process Regression (GPR)** לא היה ישים בשל **סיבוכיות חישובית של O(n3)O(n^3)O(n3)**. במקום זאת, השתמשנו ב**Nystroem Kernel Approximation** כדי להקטין את כמות החישובים מבלי לאבד את היתרונות של קרנלים. **רגרסיית Ridge** נוספה כדי למנוע **Overfitting** ולשפר את יציבות המודל.

**🔹 שלב 2: בחירת קרנלים והיפרפרמטרים**

✔ **קרנל Laplacian (degree=3, n\_components=7000)** נבחר לאחר השוואה לקרנלים אחרים (rbf, poly, sigmoid).  
✔ **רגרסיית Ridge עם α=0.05** נבחרה לאחר בדיקת ערכים שונים (0.05, 0.1, 0.5, 1.0) כדי לאזן בין דיוק המודל לבין מניעת Overfitting.

**🔹 שלב 3: אימון וביצוע Fine-Tuning**

לאחר כוונון ההיפרפרמטרים באמצעות **GridSearchCV**, אומן המודל מחדש עם הפרמטרים האופטימליים, ולאחר מכן נבדק על סט הבדיקה באמצעות מדדי ביצועים סטנדרטיים:  
✔ **R² = 0.8555** → המודל מסביר **85.55% מהשונות במחירי הטיסות**.  
✔ **MSE = 6202.68, RMSE = 78.76, MAE = 59.21** → תחזיות מדויקות, עם סטיית שגיאה ממוצעת קטנה יחסית.

**🔹 שלב 4: ניתוח השגיאות ותוצאות החיזוי**

📊 **Residual Plot (שגיאות מול מחירים חזויים):**

* רוב השגיאות מתפלגות **באופן סימטרי**, מה שמעיד על **אי-הטיה של המודל**.
* נמצאו **טעויות מעט גדולות יותר במחירי טיסות גבוהים**, מה שעשוי להעיד על שונות גבוהה יותר עבור טיסות יקרות.

📉 **Actual vs. Predicted Prices:**

* רוב החיזויים תואמים היטב את המחירים בפועל (קו אדום = חיזוי מושלם).
* קיימים מקרים בודדים של **תת-חיזוי (Under) או חיזוי-יתר (Over)**, אך במידה נמוכה.

**🔹 שלב 5: מסקנות ושיפורים אפשריים**

✅ **Kernel Approximation + Ridge Regression** מאפשר **חיזוי מחירים מדויק וניתן להרחבה (Scalable)** גם על דאטה גדול.  
✅ **קרנל Laplacian עם n\_components=7000** הצליח ללכוד את הדפוסים הלא-לינאריים של מחירי הטיסות.  
✅ **רגולריזציה (Ridge α=0.05)** מנעה Overfitting, אך עלולה לגרום לתת-חיזוי קל (Mild Underfitting).

📌 **שיפורים עתידיים:**  
1️⃣ **בדיקת ערכים נוספים של n\_components (למשל, 8000-10000) אם החומרה מאפשרת**.  
2️⃣ **הוספת פיצ’רים נוספים** כמו עונתיות, חגים, או ביקוש לפי חודשים.  
3️⃣ **ניסיון של מודלים אחרים** כמו Support Vector Regression (SVR) עם קרנלים מותאמים.

🚀 **בסיכום, המודל הותאם היטב, משיג ביצועים מצוינים, ויכול לשמש לחיזוי מחירי טיסות בדיוק גבוה!**