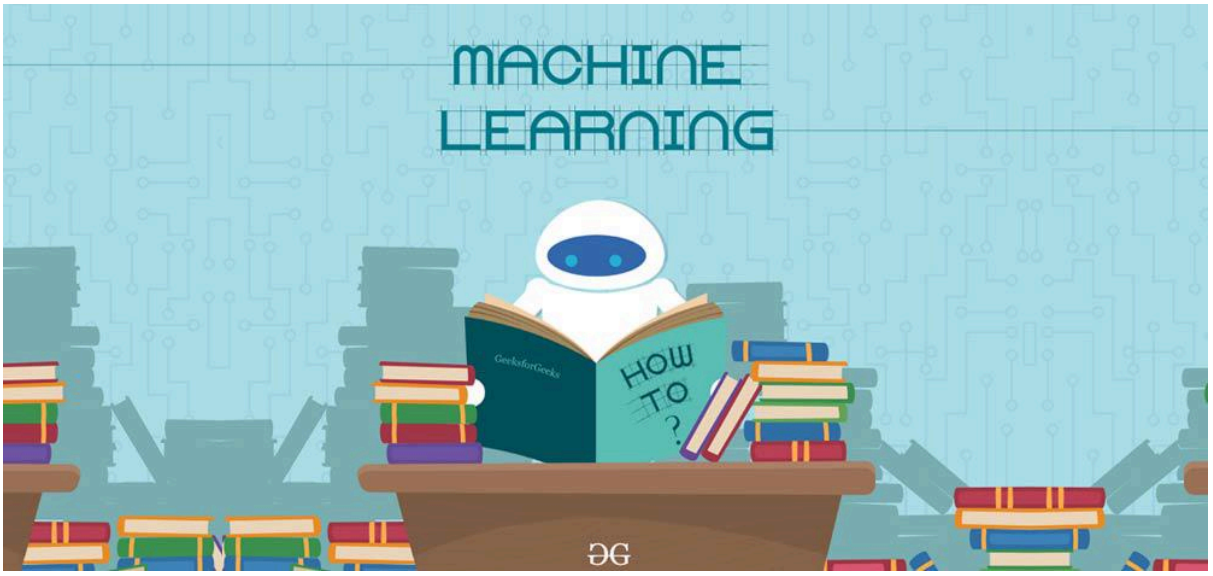
פרוייקט בלמידת מכונה

קורס מבוא ללמידת מכונה
מרצה: מר דור בנק
מתרגל: מר רון אליאס



מגישים:
גל גרינולד 211398730
אלי הלרמאיר 33791439

מבוא

בפרויקט זה התמקדנו בניתוח נתונים ובניית מודל לחיזוי תוצאות ריאיון עבודה בתכנות. המטרה הייתה להמיר את נתוני הניסיון בשפות תכנות למשתנים מספריים ולבנות מודלים שמצליחים לנבא את הסיכוי לעבור את הריאיון בהצלחה.

1. תהליך ניתוח הנתונים

1.1 טעינת הנתונים

טעינת הנתונים: הנתונים נטענו מקובץ CSV, ותיארנו את מקורם והפורמט שלהם.

1.2 בדיקה ראשונית של הנתונים:

נתונים מספריים:

על מנת להמחיש ולהבין את הנתונים הנומריים, אנו מציירים את התכונות בהיסטוגרמות וב־box plot.

- החלטנו שה-ID אינו רלוונטי לויזואליזציה.
- ניתן לראות שכ־25,722 לא עברו ו־29,740 עברו.
- אנו מבחינים ש־"A" ו־"D" מתפלגים נורמלית.
- אנו מבחינים ש־"B" ,"years_of_experience" ו־"prev_salaries" רוב הערכים מפוזרים בטווח קרוב ויש מעט ערכים קיצוניים שמושכים את החציון מטה (כפי שניתן לראות ב־boxplots). במסגרת הנדסת התכונות, נקדיש לכך תשומת לב.

תכונות שונות הן בעלות סקאלות שונות, ולכן נבחן אפשרות לנרמל אותן במסגרת הנדסת התכונות.

נתונים קטגוריאלים:

על מנת להבין את האינטראקציה של התכונות הלא נומריות עם ה־'label', נריץ גרף countplot. אנו נבחר להוציא את ה־'country' ו־'stack_experience' מהגרף כי יש להם הרבה ערכים שונים ולכן לא ניתן להמחיש אותם בגרף. נצייר את ה־'label' קודם כדי להשוות את הפיזור שלו עם התכונות האחרות.

גילינו שערכים שונים ב־'worked_in_the_past', 'disability', 'C' ו־'mental_issues' אינם משפיעים הרבה על התחזיות ויש להם פיזור דומה לזה של ה־'label'.

בנוסף, גילינו ש:

- נשים נוטות פחות לעבור.
- רוב המועמדים הם גברים.
- לא-מפתחים נוטים פחות לעבור.
- רוב המועמדים הם מפתחים.
- אנשים מבוגרים נוטים פחות לעבור מאנשים צעירים.
- סוגי השכלה שונים משפיעים בצורה שונה על הסיכוי להתקבל לעבודה.

אנו מצפים שתכונות עם אינטראקציות שונות עם ה־'label' יקבלו משקלים משמעותיים יותר במודל שלנו.

מתאם בין תכונות:

על מנת לבחון מתאם בין תכונות הרצנו heatmap על התכונות הנומריות תחילה (לאחר המרת המשתנים הקטגוריאלים נבדוק את מתאמם):

אנחנו מבחינים שיש מולטיקולינאריות בין B ו־years_of_experience. לכן, נבחן בשלב מאוחר את כדאיות השימוש בשתי התכונות.

2. עיבוד מקדים

בשלב זה נעשה מניפולציות על הדאטה שלנו לפני שלב בניית המודל.

חילקנו את הדאטה שלנו ל80% עבור אימון המודל ו20% בדיקה.

2.1 ערכים קיצוניים:

בחנו הוצאה של ערכים קיצוניים ובדקנו כיצד זה משפיע על המודל.
עבור הערכים הנומריים השתמשנו בשיטת טווחים בין-רבעוניים (IQR) עם מכפילים מותאמים אישית לכל תכונה נומרית.

- עבור כל תכונה נומרית, מחשבים את הרבעון הראשון (Q1), הרבעון השלישי (Q3), ואת ה-IQR (Q3 - Q1)).
- קבענו מכפילים לכל תכונה כדי להתאים אישית את הטווח לזיהוי ערכים קיצוניים.
- קובעים את הגבולות התחתונים והעליונים באמצעות ה-IQR והמכפיל הספציפי לתכונה (כשלא קבענו מכפיל ספציפי, ברירת המחדל היא 1.5).
- מסננים את הנתונים כך שיכללו רק את הערכים הנמצאים בתוך הגבולות המחושבים (כלומר, מסירים את הערכים הקיצוניים).

קבענו את המכפילים בצורה שונה לכל תכונה בהתאם להערכה שלנו של התפלגות התכונה - תכונות בעלות התפלגות קרובה לנורמלית יקבלו מכפיל נמוך יותר, מה שמסיר יותר ערכים קיצוניים.

<u>הנחה</u> - בשלה זה, הנחנו שהפיצ'רים הנומרים מתפלגים בצורה נומלית או חצי נורמלית (Half-normal distribution) ולכן לא אמורים להיות הרבה ערכים אשר מקובצים בקצוות (outliers).

2.2 התמודדות עם משתנים קטגוריאליים (בניית פיצ'רים חדשים):

ניסינו דרכים שונות לשנות את הערכים הקטגוריאליים למספריים ובחרנו את אלו שנותנים לנו את הvalidation_auc הגבוה ביותר עבור המודל שבנינו (פחרוט בהמשך). לבסוף בחרנו:
- עבור countries - בדקנו עבור על מדינה מה הם אחוזי המעבר שלה לקבלת העבודה ובכך נתנו לכל מדינה ציון מנורמל בין 0 ל-1, וציון זה יהיה התכונה החדשה.
- עבור stack_experience - השתמשנו בשיטה one hot encoding - כל שפת תכנות אפשרית קיבלה תכונה משל עצמה עם 0\1 בהתאם לאם השפה קיימת את מתמודד או לא. שיטה זו העלתה לנו את מספר התכונות אך **העלתה משמעותית את הvalidation_auc.**
- לשאר המשתנים הקטגוריאליים שיניו את הערכים במספרים קבועים.

- החלפת הערכים של כלל התכונות נמצאת בקוד שלנו תחת פונקציה treat_data, אותה נפעיל בהממשך גם על הtest.

2.3 נרמול ערכים:

בפונקציית treat_data ביצענו נרמול לכלל הדאטה שלנו. ניסינו דרכים שונות והשיטה שהביאה את ציון ה-auc הגבוה ביותר היר MinmaxScalar.

בדיקה סופית, הvalidation_auc יצא לנו גבוה יותר כאשר לא נירמלנו ולכן בפונקציית הtreat_data הסופית השארנו את הנירמול בcomment.

2.3 נתונים חסרים: את הערכים החסרים מילאנו בפונקציית treat_data.

את התכונות הנומריות החסרות החלפנו בעזרת knn imputation: הסבר על כך בהמשך.

מימדיות: כעת, לאחר המניפולציות על הדאטה, יש לנו 131 מימדים.

מימדיות גדולה מידיי עלולה לגרום ל: קללת המימדיות, מעלה את עלות החישוב, סיכון להתאמת יתר של המודל לדאטה שלנו.

בחנו הורדת מימדים:

בדקנו את השפעת כמות המימדים על הauc. באמצעות feature selection - forward feature selection, בפונקציה "SequentialFeatureSelector". באמצעות פונקציה זו מצאנו מה הן התכונות אשר מניבות את הauc הגבוה ביותר, אותן שמרנו בפונקציה שנשתמש בה על הtest - פונקציית "feature_selection".

את הSequentialFeatureSelector בחרנו עם ההיפר פרמטרים הבאים:

- ab_model - המודל שבחרנו.
- direction=forward - בחרנו בשיטת הforward selecting שנלמדה בכיתה.
- scoring=roc_auc - בוחרים את הפיצ'רים לפי מקסום הauc - מה שאנו מנסים למקסם בפרויקט.

לאחר הפחתת המימדים, הפער בין הtrain_auc לvalidation_auc קטן, לכן הפחתת המימדים מורידה את הoverfitting. בנוסף, בגלל שהvalidation_auc השתפר לאחר הפחתת המימדים, ונותרנו עם 64 מימדים, ניתן להניח שהמודל היה בעל מימדיות יתר ולכן בחרנו להפחית את המימדים לתכונות הטובות ביותר שמצאנו על ידי הפונקציה.

החלת העיבוד המקדים על סט ה-Test:

הורדנו את הדאטה של הtest וביצענו עליו את המניפולציות שבחרנו ושפירטנו עליהן לעיל (על ידי הפעלת פונקציות).

ניסיונות אשר לא נלמדו בקורס:

בניית פיצ'רים - stack_experience בשיטת one hot encoding.

הוצאת outliers על ידי שיטת טווחים בין רבעוניים - IQR.

3. הרצת המודלים

בשלב זה התאמנו לtraining_data שלנו מודלים שונים:

- פירוט על ההיפר הפרמטרים השונים שבחנו במודלים, והשפעתם על שונות והטיית המודל, מפורטים בנספח "למידת מכונה - נספח היפרפרמטרים".

3.1 Logistic Regression

בנינו פונקציה find_best_max_iter אשר בוחרת את הפרמטר הכי טוב לmax_iter בין 100 ל1,000 בקפיצות של 100. הפונקציה משתמשת בGridSearchCV ובוחרת את ההיפר-פרמטר שנותן את הביצועים הטובים ביותר בcross-validation, (היפרפרמטר cv=5).

3.2 Gaussian Naive Bayes- מודל זה לא מקבל היפר-פרמטרים.
על מנת להישתמש במודל יש להניח את ההנחות הבאות:

- הפיצ'רים של הדאטה שלנו מתפלגים נורמלית.
- הפיצ'רים של הדאטה אינם תלויים.

3.3 Random forest- במודל זה ניסינו היפר-פרמטרים שונים ובחרנו באלו שממקסמים לנו את הvalidation_auc (השתמשנו בparam_grid).

3.4 AdaBoost- בחרנו את ההיפר-פרמטרים במודל זה באמצעות הרצת Grid Search.
מבין ארבעת המודלים, הAdaBoost הניב לנו את התוצאה הטובה ביותר.

תרומת הפיצ'רים
עבור ארבעת המודלים הצגנו תרשימים של חשיבות הפיצ'רים. עבור AdaBoost וRandom Forest יש פונקציה מובנית - plot_importance, שמטרתה להציג את השפעת הפיצ'רים לפי גודל חשיבות על המודל שלנו. עבור המודלים Logistic Regression וGaussian Naive Bayes בנינו פונקציה שתראה עבורם את תרומת הפיצ'רים גם כן. הצגנו את התרשימים (תרשימים מצורפים בנספח התרשימים).

* עמודת הnan מראה 1 עבור משתתפים stack_experience שלהם היה ריק (לאחר שביצענו one hot encoding). החלטנו ללהשאיר את פיצ'ר ה-nan משום שזה הניב לנו את התוצאות הטובות ביותר.

ניתן להבחין שהפיצרים שבנינו משפות התכנות של הstack_experience הן בעלות ההשפעה הגבוהה ביותר. זוהי אחת הסיבות שהקדשנו מאמץ רב בביצוע מניפולציות שונות על פיצ'ר זה, מה שבסופו של דבר הביא אותנו לשיטת הone hot encoding, אשר הקפיצה לנו את תוצאות המודל.

4. הערכת המודל- בשלב זה ביצענו ניתוחי הערכה שונים על המודלים שלנו.

4.1 Confusion Matrix

ביצענו תחזיות על נתוני האימון של הדאטה והצגנו את התוצאות בצורת confusion matrix. (תרשים מצורף בנספח התרשימים).

שליליים אמיתיים: 18,351 אלה הם האנשים שנחזו נכון להיכשל בראיון התוכנה. מספר גבוה זה מצביע על כך שהמודל טוב בזיהוי הנכשלים.

חיוביים כוזבים: 1532 אלה הם האנשים שנחזו להצליח בראיון התוכנה אך בפועל נכשלו. מספר זה מצביע על כך יכולת המודל לבצע תחזיות חיוביות שגויות.

שליליים כוזבים: 776 אלה הם האנשים שנחזו להיכשל בראיון התוכנה אך בפועל הצליחו. מספר נמוך יחסית זה מצביע על כך שהמודל טוב במזעור מקרים חיוביים שהוחמצו.

חיוביים אמיתיים: 22,406 אלה הם האנשים שנחזו נכון להצליח בראיון התוכנה. מספר גבוה זה מצביע על כך שהמודל טוב בזיהוי המצליחים.

מסקנות:

שיעורי חיוביים אמיתיים ושליליים אמיתיים גבוהים: הערכים הגבוהים בתאים האלכסוניים (TN ו-TP) מראים שמודל ה-Random Forest בעל דיוק גבוה בחיזוי מקרים של הצלחה וכישלון.

שיעורי חיוביים כוזבים ושליליים כוזבים נמוכים: הערכים הנמוכים יחסית בתאים הלא-אלכסוניים (FP ו-FN) מצביעים על כך שהמודל מבצע מעט שגיאות בשני סוגי התחזיות השגויות.

4.2 K-Fold Cross Validation- בשלב זה ביצענו הערכת מודל באמצעות Cross Fold-K Validation ובנינו פלט ROC על כל אחד מהמודלים (תרשים מצורף בנספח התרשימים).

מתרשים זה ניתן להבחין שמבין ארבעת המודל שלנו, המודל שמגיע לAUC הגבוה ביותר הוא הAdaBoost 0.98. לכן זה המודל שנבחן עבור ביצוע התחזיות בtest.

4.2 פערי ביצוע בין הTrain לValidation- על מנת לבחון את פערי הביצוע בין הTrain לValidation של המודלים שלנו בנינו את הפונקצייה compare_validation_training_gap. אשר מקבלת את train_data ואת validation_data ומדפיסה את הAUC של כל דאטה לכל מודל, ואת ההפרש בינהם. התוצאות מצורפות בנספח תרשימים.

ניתן להבחים שהפרשי הAUC מאוד קטנים.

בAdaBoost ההפרש שלילי (כלומר בvalidation auc גבוה מהtrain_auc). משום שההפרש הוא מאוד קטן (פחות מאלפית לAdaBoost, החלטנו שההפרש הוא מזערי ושייתכן משום שהמודל הוא בעל AUC גבוה מאוד - סביב ה0.984.

מצד שני ברגרסיה לוגיסטית קיבלנו גם הפרש שלילי, רק שהפעם הauc הוא סביב ה0.535. כלומר מודל זה מניב לנו תוצאות רעות, ולא מומלץ להסתמך עליו.

בRandom Forest וNaive Bayes ההפרש חיובי אך קטן.

גם בRandom Forest וגם בAdaBoost, הtraining AUC מאוד גבוה, לכן חשדנו בoverfitting, אך משום שvalidation AUC גם מאוד גבוה וההפרש קטן ניתן להניח שאנו לא נמצאים בהתאמת יתר.

5. ביצוע פרדיקציה- לאחר שביצענו עיבוד מקדים לסט המבחן, הרצנו predict בעזרת המודל הטוב ביותר שבנינו, ושמרנו את הסתברויות הסיווג לצד תעודות הזהות בקובץ csv בשם results_27.

בסוף הקובץ, יש תיבת קוד "pipeline" אשר ניתן בעזרתה להריץ את מלאו הקוד ולבצע חיזוי.

6. שימוש בכלים שלא נלמדו בקורס

שימוש ב-KNN imputation בפרויקט-

אתחול: קביעת מספר שכנים (n_neighbors).
חישוב מרחקים: חישוב מרחקים בין דגימות.
אימפוטציה: החלפת ערכים חסרים בממוצע הערכים של השכנים הקרובים.
בחרנו בשיטה זו למילוי ערכים חסרים כדי לשמר על הקשרים בין התכונות, מה שהביא לשיפור באיכות המודל.

שימוש ב-Meta-Learning בפרויקט-
אתחול: קביעת פרמטרים והגדרת מספר ניסיונות (n_trials).
אופטימיזציה: האלגוריתם משתמש בתוצאות הקודמות כדי לשפר את ההמלצות בפרמטרים של האיטרציות הבאות ובכך מתכנס מהר יותר לפרמטרים הטובים ביותר.
אימון מודל: בחירת הפרמטרים האופטימליים ואימון המודל על פיהם. בחרנו בשיטה זו כדי למצוא את הפרמטרים המיטביים באופן יעיל יותר מאשר חיפוש רשת מסורתי, מה שהביא לשיפור מדדי הביצועים של המודל.

נספח - חלוקת אחריות

גל גרינולד 211398730
אלי הלרמאיר 33791439

את חלוקת העבודה בפרויקט חילקנו בצורה שוויונית.
שיטת העבודה שלנו הייתה להיפגש למפגשים שבועיים גדולים, בסוף כל מפגש חילקנו משימות לשבוע, כאשר באחריות כל אחד היה לסיים אותן עד למפגש הבא.

אלי הלרמאיר פעל יותר בשיפור המודלים ומקסום הAUC ורב המשימות האישיות שלו התמקדו בכך.
גל גרינולד התמקד יותר בהבנת הדאטה וניתוחו, במקביל להבנת הפרויקט ומיקודו אחד ההנחיות המדויקות.

לאורך הפרויקט פעלנו בצורה משותפת וכל שינוי של הקוד או של התיעוד של אחד מחברי הקבוצה היה בשיתוף החבר האחד.
העבודה המשותפת תרמה לנו והייתה מפתחת. הרבה פעמים סיעור מוחות היה הדרך היעילה עבורנו להגיע לפתרון אשר היה בעיננו אופטימלי.

נספח - פירוט על ההיפר-פרמטרים של המודלים, והשפעתם על השונות וההטיה

גל גרינולד 211398730
אלי אלנמהייר 33791439

Logistic Regression

הפרמטר max_iter שולט על מספר האיטרציות המרבי שהאלגוריתם מורשה לרוץ במהלך תהליך האופטימיזציה.
max_iter גבוה: מפחית הטיה אך עלול להגדיל שונות ולהוביל ל-overfitting.

Random forest

פרמטרים:

- n_estimators - מספר העצים ביער.
- entropy - בודק את תוספת המידע כדי להנחות את תהליך קבלת ההחלטות של העצים.

יותר עצים בדרך כלל משפרים את הביצועים אך דורשים יותר חישוב.
יותר עצים (n_estimators גבוה יותר) מפחיתים שונות ויש להם השפעה מועטה על ההטייה.
במודל זה ניסינו היפר-פרמטרים שונים ובחרנו באלו שממקסמים לנו את הvalidation_auc.

AdaBoost

פרמטרים:

- base_estimator: התת-מודל של העץ המשמש את מודל הבוסטינג:
  - max_depth - עומק העץ.
  - min_samples_split - המספר המינימלי של נתונים שצריכים להיות בעלה על מנת שיוכל להתפצל.
- n_estimators: מספר תתי-העצים שישולבו.
- learning_rate: גודל ה"צעדים" של המודל - כמה כל לומד חלש תורם למודל.

השפעה על שונות והטייה:

max_depth ו-min_samples_split (עבור DecisionTreeClassifier):

- max_depth גבוה יותר: מפחית הטייה, מעלה שונות.
- min_samples_split גבוה יותר: מעלה הטייה, מפחית שונות.

n_estimators - גבוה יותר (יותר עצים), מפחית שונות ויש לו השפעה מועטה על ההטייה.

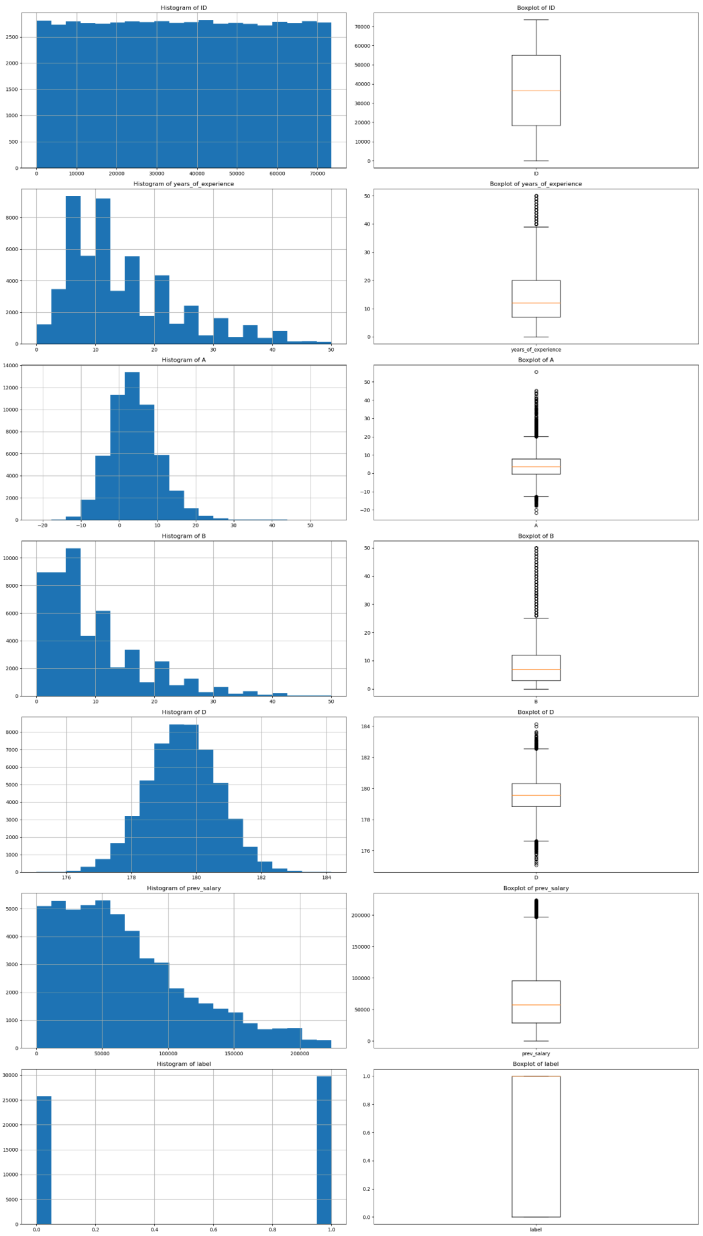
learning_rate - נמוך יותר: מפחית שונות, אך דורש יותר לומדים חלשים כדי להשיג את אותה רמת ביצוע.

נספח תרשימים

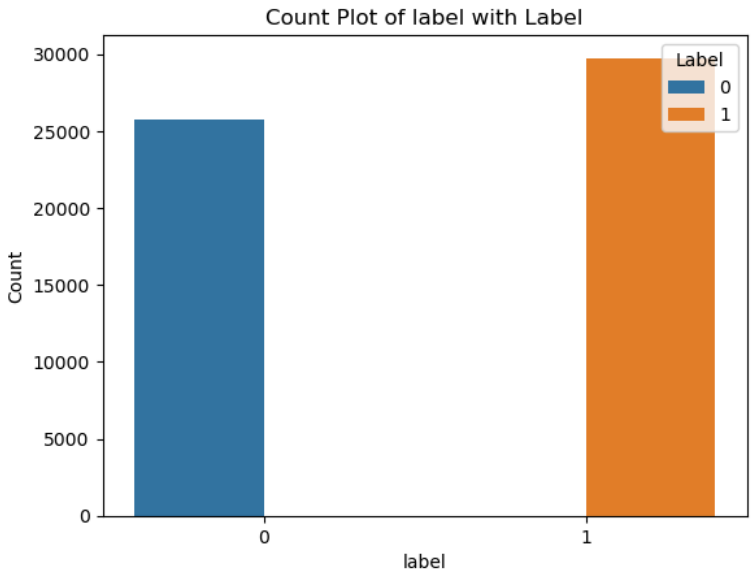
גל גרינולד 211398730
אלי הלרמאיר 33791439

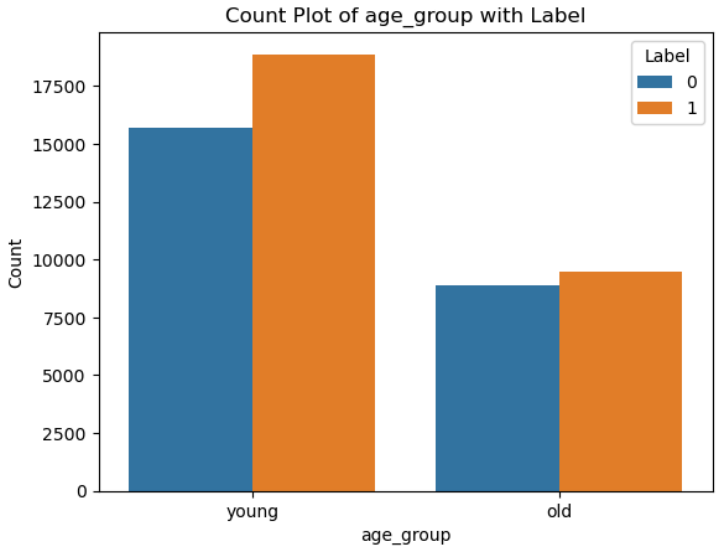
א. תרשימי תהליך ניתוח הנתונים

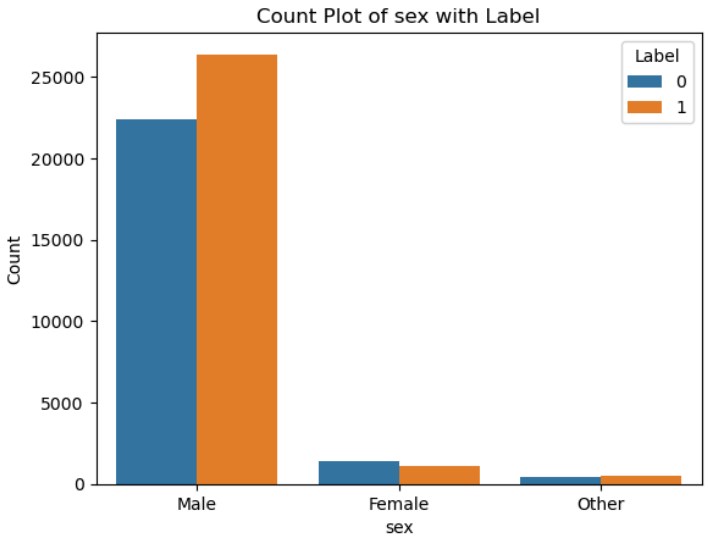
נומריים:

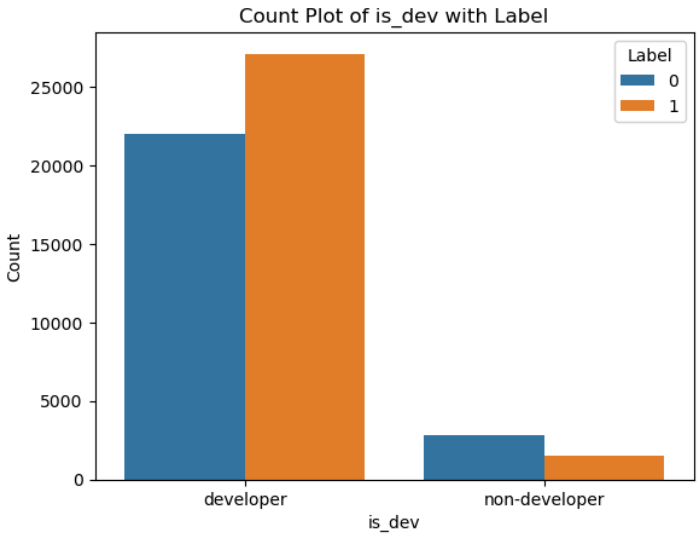


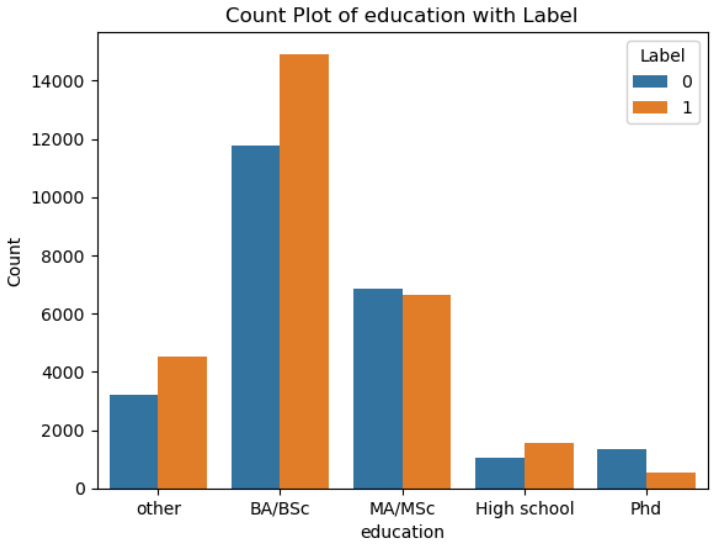
קטגוריאליים:

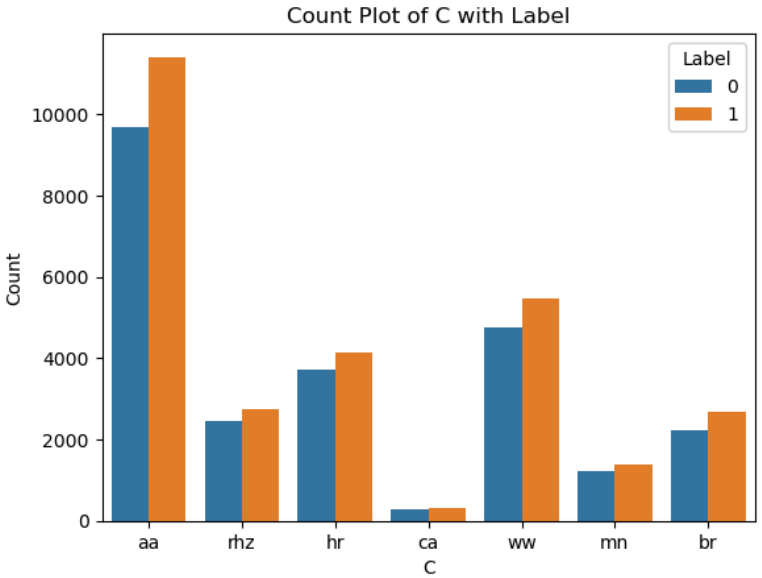


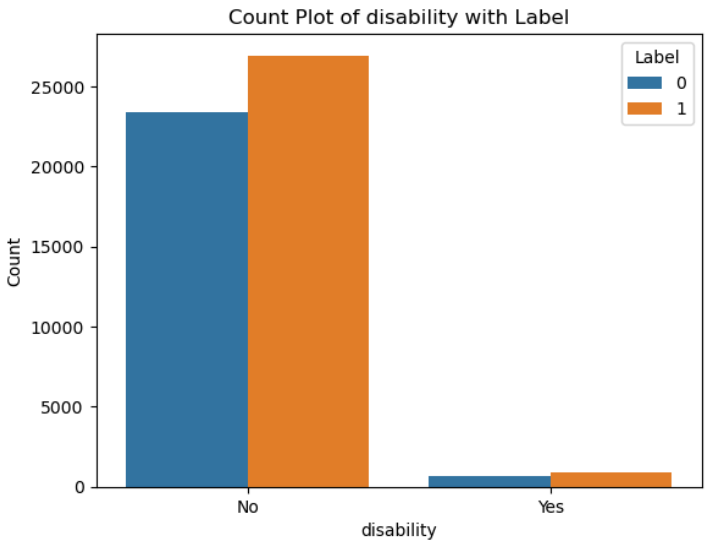


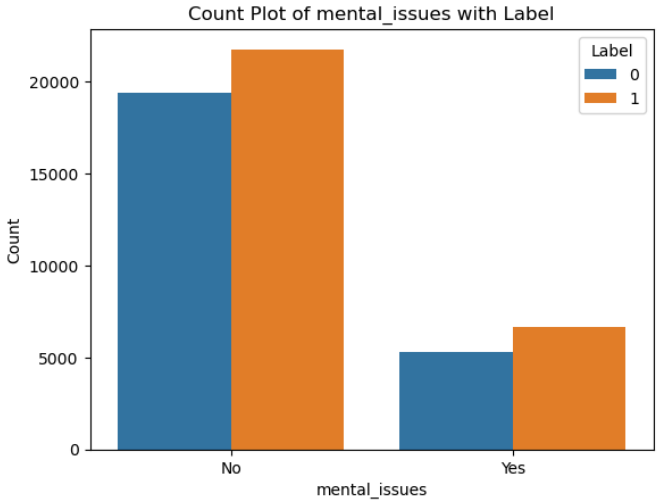


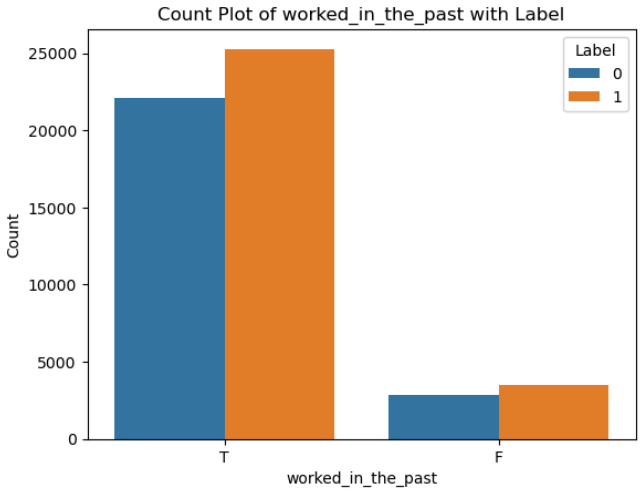

Count Plot of is_dev with Label
Label
0
1
Count
25000
20000
15000
10000
5000
0
developer
non-developer
is_dev


Count Plot of education with Label
Label
0
1
Count
14000
12000
10000
8000
6000
4000
2000
0
other
BA/BSc
MA/MSc
High school
Phd
education

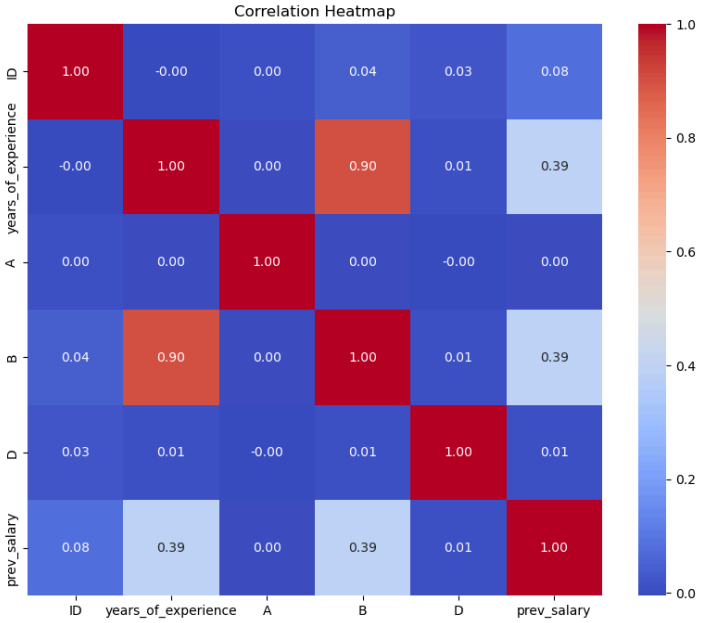

Count Plot of C with Label
Label
0
1
Count
10000
8000
6000
4000
2000
0
aa
rhz
hr
ca
ww
mn
br
C


Count Plot of disability with Label
Label
0
1
Count
25000
20000
15000
10000
5000
0
No
Yes
disability

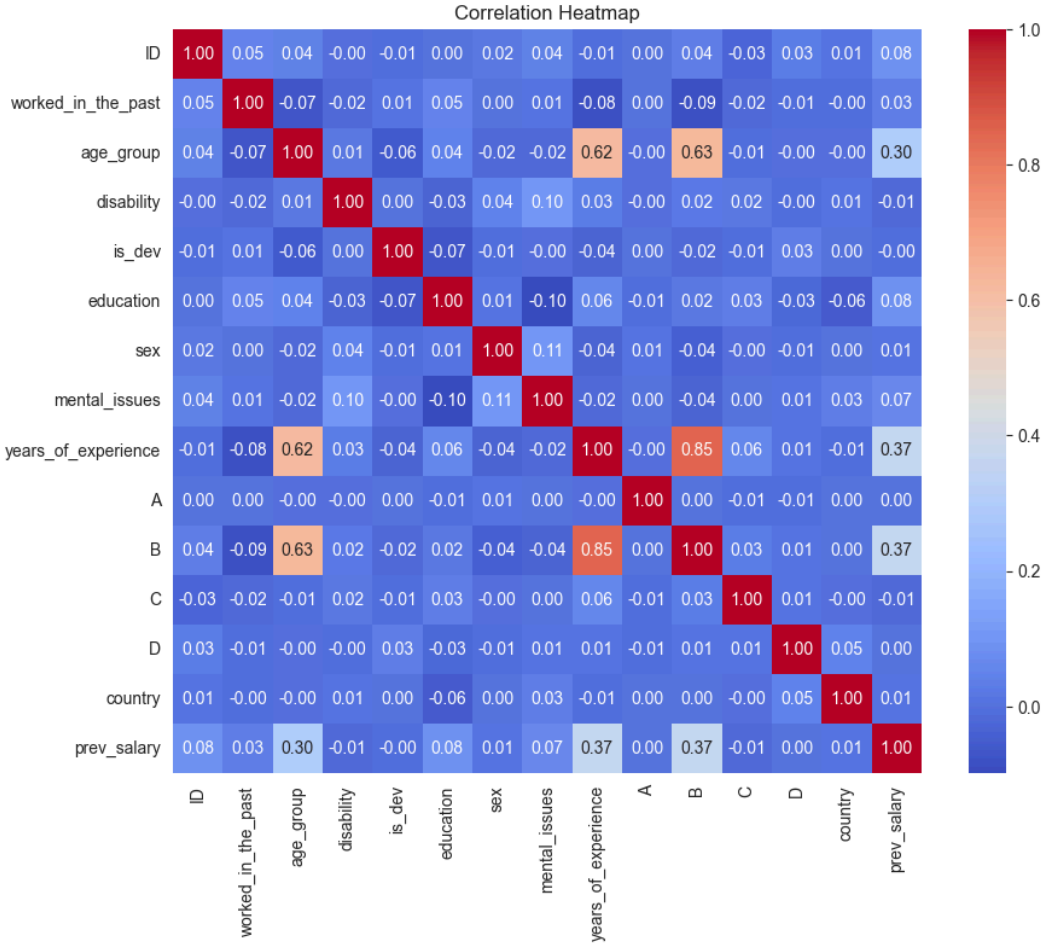

Count Plot of mental_issues with Label
Label
0
1
Count
20000
15000
10000
5000
0
No
Yes
mental_issues


Count Plot of worked_in_the_past with Label
Label
0
1
Count
25000
20000
15000
10000
5000
0
T
F
worked_in_the_past

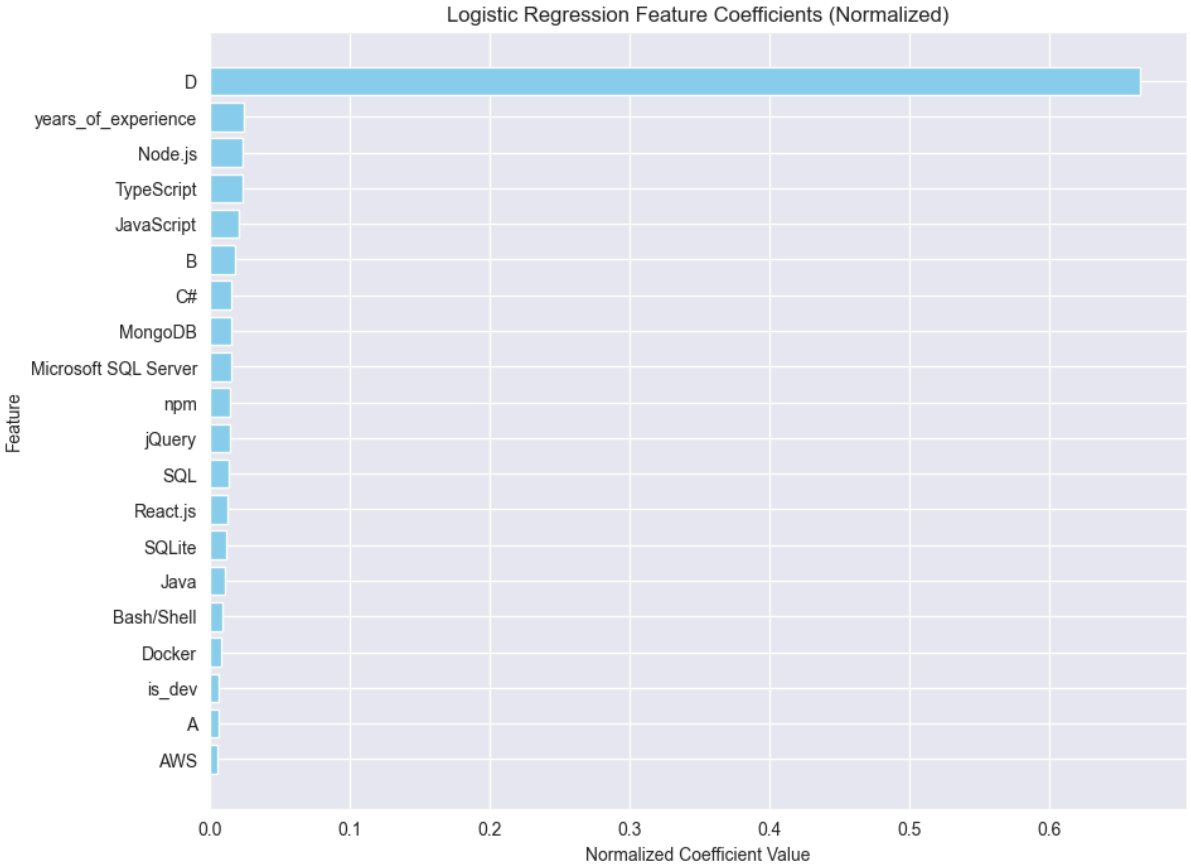
מתאם ערכים נומריים:

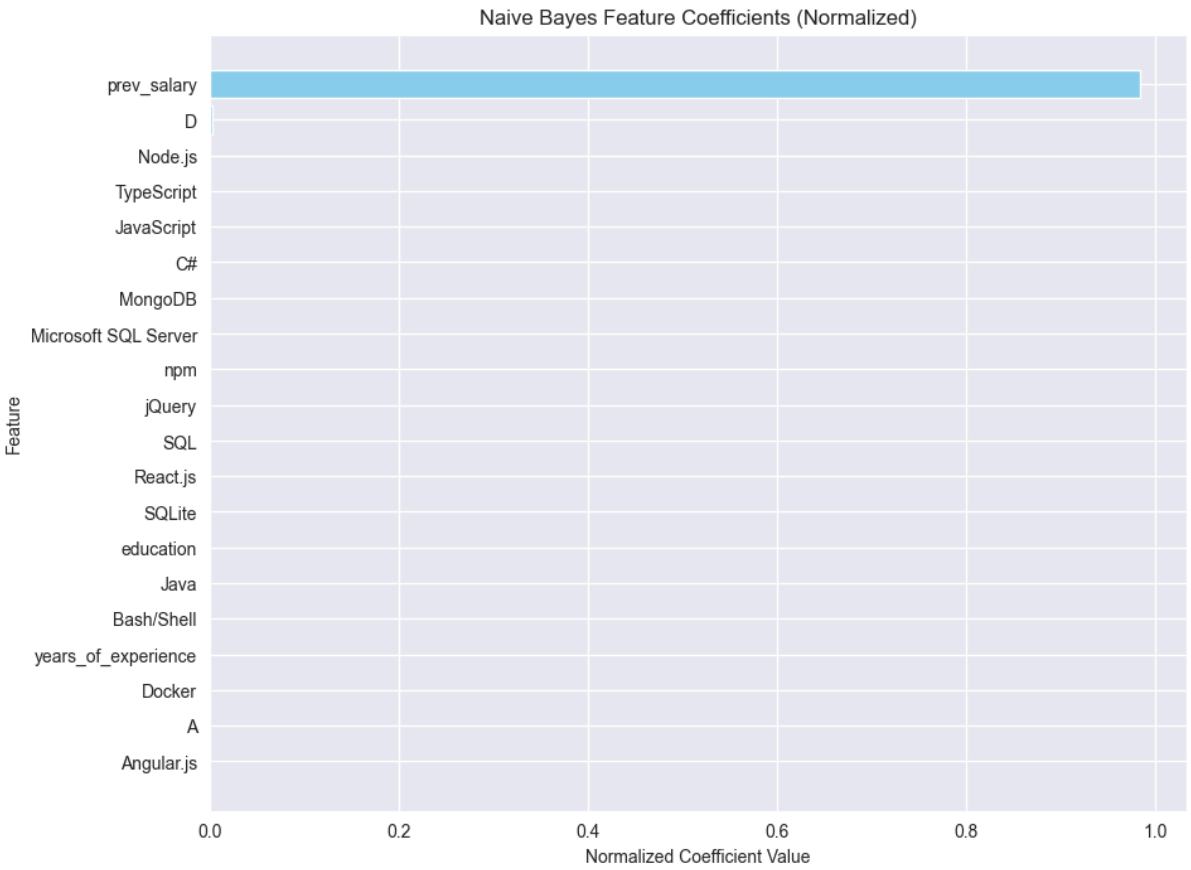


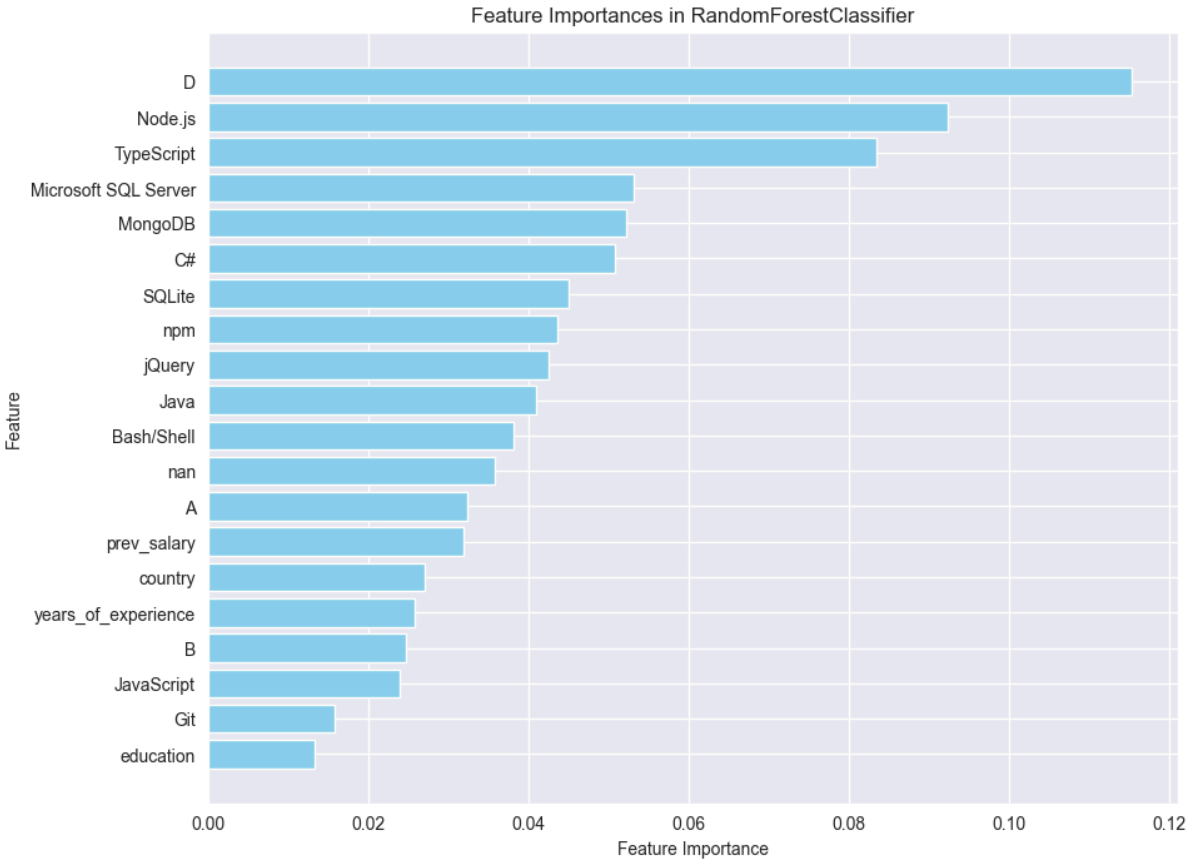
מתאם לאחר המרת הערכים הקטגוריאלים לנומריים:

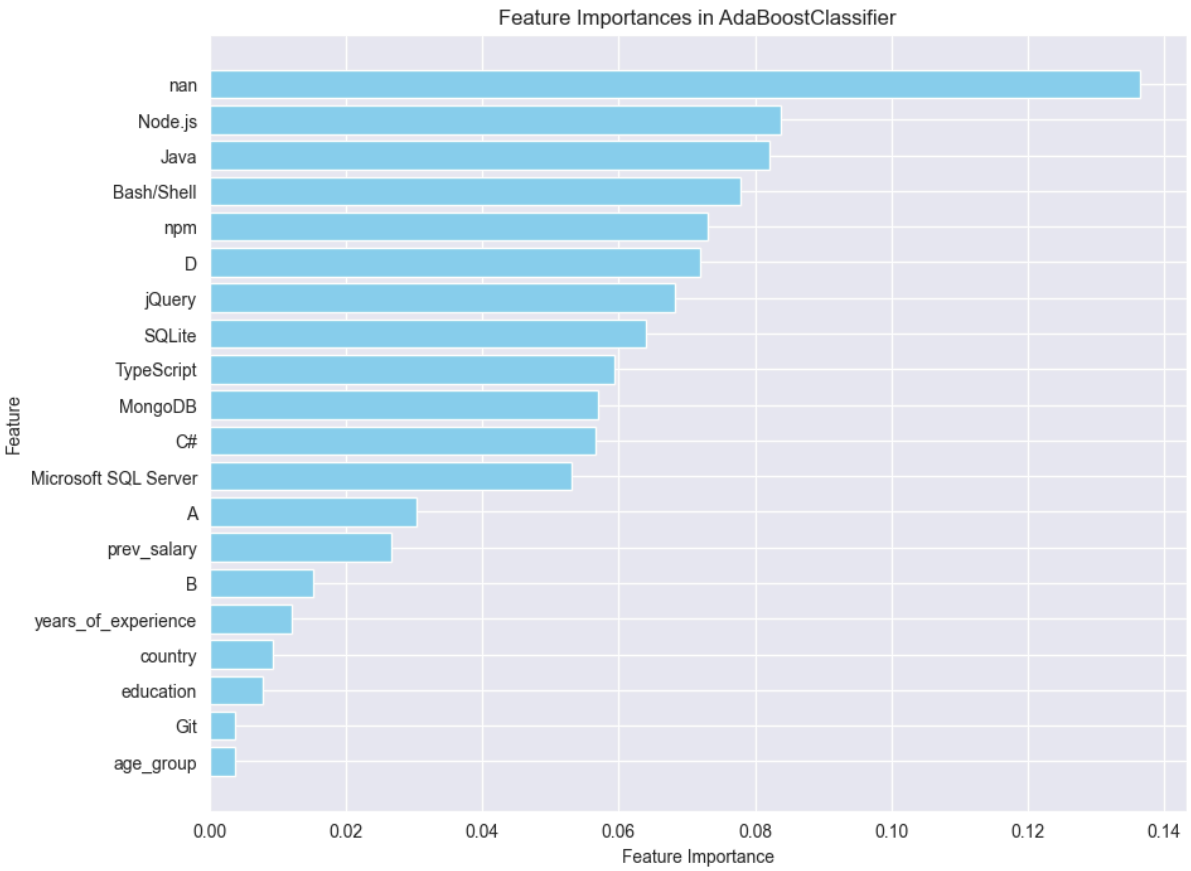


ב. תרשים תרומת הפיצ'רים:



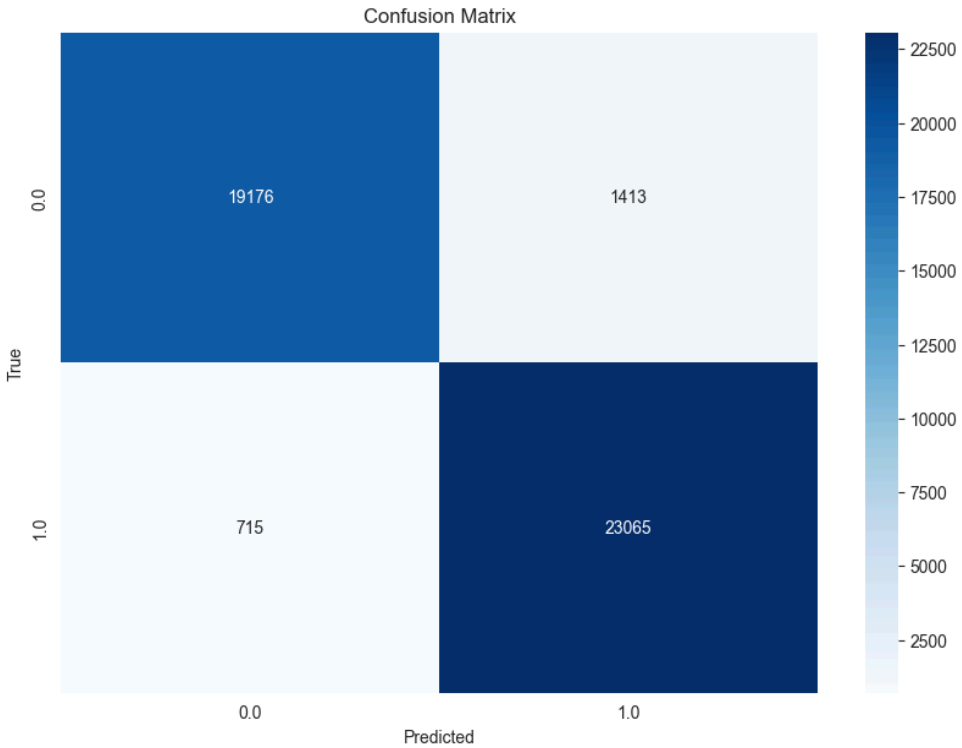

Naive Bayes Feature Coefficients (Normalized)
prev_salary
D
Node.js
TypeScript
JavaScript
C#
MongoDB
Microsoft SQL Server
npm
jQuery
SQL
React.js
SQLite
education
Java
Bash/Shell
years_of_experience
Docker
A
Angular.js
Feature
0.0
0.2
0.4
0.6
0.8
1.0
Normalized Coefficient Value


Feature Importances in RandomForestClassifier
D
Node.js
TypeScript
Microsoft SQL Server
MongoDB
C#
SQLite
npm
jQuery
Java
Bash/Shell
nan
A
prev_salary
country
years_of_experience
B
JavaScript
Git
education
Feature
0.00
0.02
0.04
0.06
0.08
0.10
0.12
Feature Importance

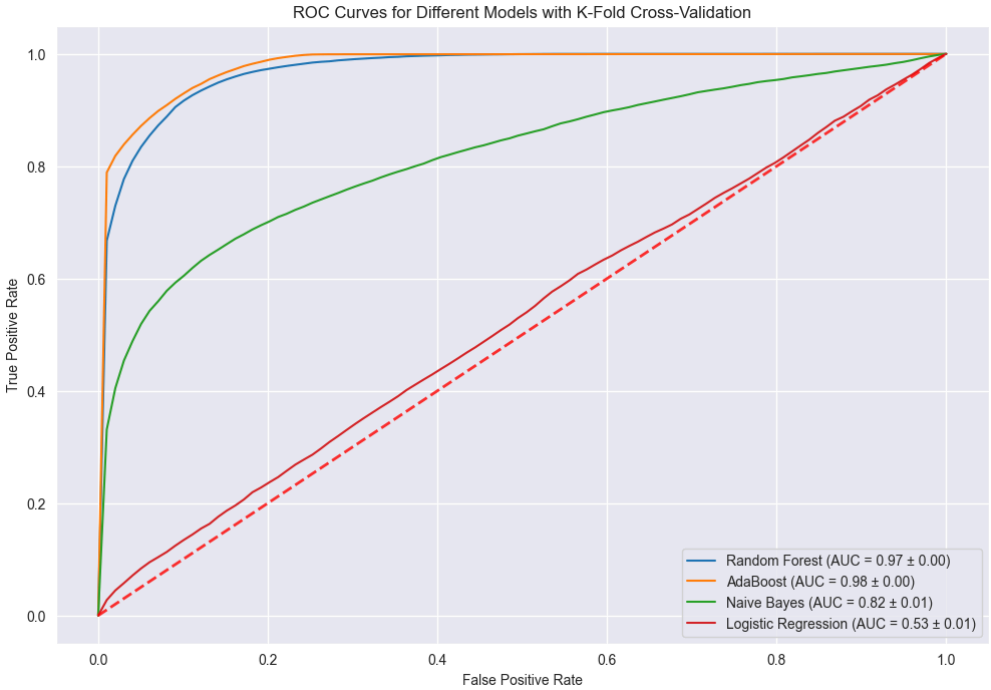


ג. הערכת מודלים:

Confusion Matrix



K-Fold Cross Validation



פערי ביצוע בין הTrain לValidation

	Model	Train AUC	Validation AUC	AUC Gap
0	Random Forest	0.994727	0.975916	0.018811
1	AdaBoost	0.983939	0.984009	-0.000070
2	Naive Bayes	0.819414	0.816751	0.002663
3	Logistic Regression	0.535481	0.543881	-0.008400