פרוייקט גמר
ביג דאטה בתעשייה
אישור הלוואות אשראי

מרצה:
אריאל בר

מגישים:
דניאל *****
נופר *****

לינק לקאגל: https://www.kaggle.com/datasets/ppb00x/credit-risk-customers

קריאה מהנה 😊

מטרת הפרוייקט:

מטרת פרויקט זה היא לפתח מודל סיווג מדויק הקובע האם כדאי לאשר בקשת הלוואה של לקוח על סמך יכולתו להחזיר את ההלוואה. המודל ינתח מאפיינים שונים המייצגים מידע על הלקוח ויחשב את ההסתברות להחזר הלוואה מוצלח. הפלט הסופי יהיה סיווג בינארי המציין האם יש לאשר או לדחות את ההלוואה.

לאור השימוש הנרחב ב-AI בתחום קבלת החלטות פיננסיות, פרויקט זה נועד למנף טכניקות למידת מכונה מתקדמות כדי לשפר את הדיוק והאמינות של החלטות אישור הלוואות. על ידי שימוש בטכניקות שונות של feature engineering, שיטות ensemble, רשתות ניורנים וכוונון hyperparameters, המודל שואף לתפוס דפוסים מורכבים בנתונים ולספק את התובנות החשובות לגבי התנהגות הלקוחות ויכולתם להחזיר את ההלוואה.

פרויקט זה רלוונטי במיוחד במגזר הפיננסי, שבו תחזיות מדויקות יכולות להפחית סיכונים ולשפר תהליכי קבלת החלטות. כידוע הבנקים היום (נוכל לקחת את ארה"ב כדוגמה) מחלקים הלוואות להמון לקוחות והרבה מהם לא מצליחים להחזיר את החובות. נוכל לקחת את הקריסה של הבנק הענק Washington Mutual שחילק משכנתאות בארה"ב בצורה אגרסיבית מאוד וכדובדבן על הקצפת כל זה קרה לפני קריסת שוק ההון בשנת 2008, מה שגרם לבנק לקרוס בצורה טוטאלית.

מבחינת מקרים מסוג ניתן לראות את חשיבות הפרוייקט שלנו, במיוחד בתקופה שלנו שיש לנו את הכלים הדרושים כדי לקבל החלטות תבוניות.

מודלים רבים של AI הוכיחו את יעילותם בבעיות סיווג דומות, והוכיחו את היתרונות הפוטנציאליים של יישום למידת מכונה על תרחישי אישור הלוואות.

נתונים:

הדאטה שנעבוד איתו הינו דאטה מסוג Tabular (טבלאי) שמורכב **מרשומות שכתובות בצורה ש"בני אדם" כותבים**. כל הפרטים הם עבור כל לקוח המבקש הלוואה, וחשוב לציין שהמידע בצורה הקיימת הוא מאוד לא נוח לעבודה ומקשה על המודלים הקיימים ועל כן דרוש Preprocess וFeature engineering מאוד מורכבים.
קישור לדאטהסט ניתן למצוא בלינק הבא: credit risk customers

מבט על הדאטה:

- הנתונים נאספו בגרמניה. הנתונים מייצגים בשקיפות מלאה את המציאות, כלומר המידע נלקח מבנק בגרמניה עם המידע שהוכנס למערכת בתהליך קבלת ההחלטה והתיוג ולבסוף הוא משקף את ההחלטה שהתקבלה על ידי צוות הבנק.
- הסט נתונים מורכב מ1000 רשומות כאשר בכל אחת מהרשומות קיימים 20 פיצ'רים. חשוב לשים לב לפרט זה מכיוון שבמידה ונרצה להרחיב לרשת מורכבת 1000 רשומות אינן מספיקות וצריך לייצר דאטה נוסף.
- ההתפלגות בסט הנתונים הזה אינה מאוזנת כאשר קיימות 700 רשומות עם תיוג "לתת הלוואה" ו300 רשומות "לא לתת הלוואה". בהמשך נוכל לראות את הגרף עבור זה.
- 13 פיצ'רים קטגוריילים ו7 פיצ'רים נומריים.

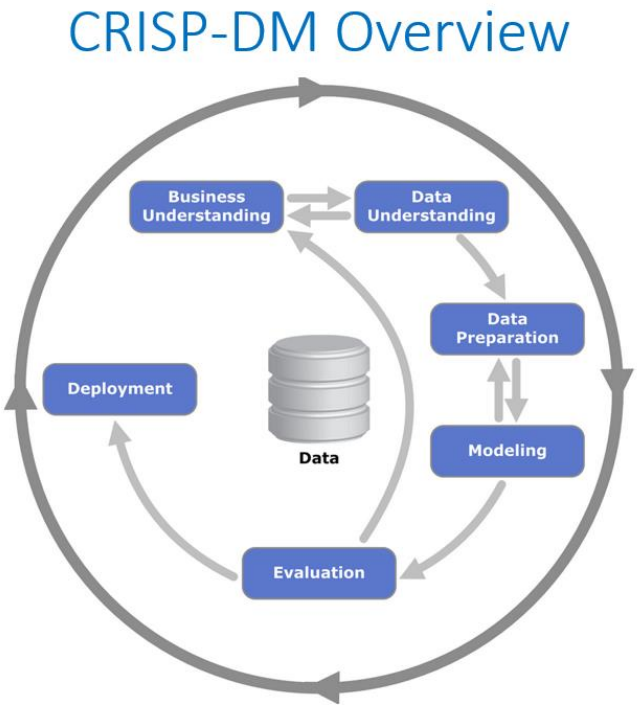
```
[ ] df.head()
```

[4]

checking_status	duration	credit_history	purpose	credit_amount	savings_status	employment	installment_commitment	personal_status	other_parties	...	property_magnitude	age	other_payment_plans	housing	existing_credits	job	num_dependents	own_telephone	foreign_worker	class
<0	6.0	critical/other existing credit	radio/tv	1169.0	no known savings	>=7	4.0	male single	none	...	real estate	67.0	none	own	2.0	skilled	1.0	yes	yes	good
0<=X<200	48.0	existing paid	radio/tv	5951.0	<100	1<=X<4	2.0	female div/dep/mar	none	...	real estate	22.0	none	own	1.0	skilled	1.0	none	yes	bad
no checking	12.0	critical/other existing credit	education	2096.0	<100	4<=X<7	2.0	male single	none	...	real estate	49.0	none	own	1.0	unskilled resident	2.0	none	yes	good
<0	42.0	existing paid	furniture/equipment	7882.0	<100	4<=X<7	2.0	male single	guarantor	...	life insurance	45.0	none	for free	1.0	skilled	2.0	none	yes	good
<0	24.0	delayed previously	new car	4870.0	<100	1<=X<4	3.0	male single	none	...	no known property	53.0	none	for free	2.0	skilled	2.0	none	yes	bad

בעצם הסט נתונים הזה מאופיין בחוסר סדר בתוכו, אך הבעיה שקיימת כאן היא משמעותית מאוד וקריטית מאוד לעולם האמיתי. יותר מכך עד כה בכל הקורסים שלנו לאורך התואר לא נתקלנו במטלה שדורשת Preprocess וFeature engineering כלכך מורכבים שדורשים חשיבה והבנה של הנתונים והתחום. על כן העבודה דרשה בעיקר המון ניסוי וטעיה עם הפיצ'רים שבעצם נתנה לנו ליישם את כל העקרונות של CRISP-DM , העבודה התאפיינה בכך שהתחלנו בהתחלה בלנרמל את הנתונים ולסדר אותם והתפתחה לכך שהיינו יוצרים כמות של פיצ'רים ומנסים אותם על המודלים שלנו, לאחר קבלת התוצאות חזרנו אחורה ויצרנו עוד פיצ'רים ובעזרת SHAP גם הבנו את הפיצ'רים הפחות משמעותיים והורדנו אותם על מנת שלא נגיע גם למצב של Overfit .

כפי שצוין לעיל, pipeline הפרוייקט שלנו בוצע על מתודולוגית CRISP-DM:



ניתוח מקדים:

1. Data understanding –

בשלב הראשון של הבנת הנתונים, חקרנו את הדאטה והמשמעויות של כל תכונה בו, במטרה להבין את המשמעות של כל פיצ'ר וכיצד הוא משפיע על הסיכון קבלת ההלוואה של הלקוח. הבנה מעמיקה זו מאפשרת לנו בחירה נבונה יותר של הפיצ'רים שישמשו את המודלים בהמשך. ניתן לראות כדוגמא פיצ'ר כמו other_payment_plans המכיל מידע על תכניות תשלום נוספות שיש ללקוח. תכניות אלו עשויות להיות הסדרי אשראי נוספים שהלקוח התחייב להם, כמו חלוקה לתשלומים בחנויות בגרמניה, אשר נתפסת כהלוואה נוספת ושונה מהלוואה בנקאית רגילה. הבנה של ההבדל הזה מסייעת בהערכת עומס החובות של הלקוח ובקביעת הסיכון נוסף הנובע ממנו.

נוסף על כך, חקרנו פיצ'רים נוספים כמו checking_status, שמתאר את מצב חשבון העובר ושב של הלקוח ומספק תובנות על מצבו הכלכלי הנוכחי, ו-credit_history, המתאר את היסטוריית האשראי של הלקוח ומספק מידע על אופן ההתנהלות הפיננסית שלו בעבר. לדוגמה, לקוח עם credit_history המציין בעיות משמעותיות באשראי קודם עשוי להיחשב כסיכון גבוה יותר בהשוואה ללקוח עם היסטוריה של תשלומים מוצלחים.

באופן דומה, הבנת המשמעות של הפיצ'רים הקשורים למטרת האשראי (purpose), למצב החיסכון (savings_status), ולמצב התעסוקה (employment) אפשרה לנו להעריך בצורה טובה יותר את יכולתו של הלקוח להחזיר את ההלוואה ואת הסיכונים הפיננסיים הכרוכים בכך.

ניתוח מעמיק של כל הפיצ'רים מאפשר זיהוי של דפוסים חשובים שיכולים להנחות את תהליך בחירת הפיצ'רים הסופי ולהוביל לתוצאות מדויקות יותר במודלים החזויים.

2. ניתוח נתונים ראשוני –

בשלב זה, כפי שניתן לראות בתצלום מטה, השתמשנו בפונקציה describe. פונקציה זו אפשרה לנו לקבל מושג ראשוני על הסטטיסטיקות של המשתנים הנומריים בדאטה. באמצעות describe, ניתחנו מדדים בסיסיים כמו ממוצע, חציון, סטיית תקן, והערכים המקסימליים והמינימליים של כל משתנה. מידע זה חיוני כדי להבין את ההתפלגות של כל משתנה, לזהות חריגים פוטנציאליים, ולהעריך את טווח הערכים השכיחים עבור כל תכונה.

לדוגמה, ניתוח ממוצע הערכים וסטיית התקן של משתנים כמו duration ו-credit_amount מסייעים בהבנת הרמות הכלליות של אשראי ומשך ההלוואות בקרב הלקוחות. יתר על כן, בדיקת הערכים המקסימליים והמינימליים מאפשרת לנו לזהות אם קיימים ערכים חריגים או בלתי סבירים שעלולים להשפיע על המודלים החזויים בהמשך.

באמצעות ניתוח ראשוני זה, קיבלנו הבנה טובה יותר של מבנה הנתונים הנומריים, מה שמאפשר לנו להמשיך לשלב הבא בתהליך הניתוח עם הבנה מבוססת של תכונות המפתח וההתפלגות שלהן.

```
df.describe()
```

	duration	credit_amount	installment_commitment	residence_since	age	existing_credits	num_dependents
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	20.903000	3271.258000	2.973000	2.845000	35.546000	1.407000	1.155000
std	12.058814	2822.736876	1.118715	1.103718	11.375469	0.577654	0.362086
min	4.000000	250.000000	1.000000	1.000000	19.000000	1.000000	1.000000
25%	12.000000	1365.500000	2.000000	2.000000	27.000000	1.000000	1.000000
50%	18.000000	2319.500000	3.000000	3.000000	33.000000	1.000000	1.000000
75%	24.000000	3972.250000	4.000000	4.000000	42.000000	2.000000	1.000000
max	72.000000	18424.000000	4.000000	4.000000	75.000000	4.000000	2.000000

בנוסף, ביצענו סיכום של הנתונים הקטגוריאליים על מנת ללמוד ולהסיק תובנות חשובות להמשך הניתוח. סיכום זה אפשר לנו להבין את הקרדינליות של המשתנים הקטגוריאליים ולהבחין בין עמודות עם שונות גבוהה לבין עמודות עם ערכים מוגבלים.

<u>תובנות מהפלטים שהתקבלו:</u>

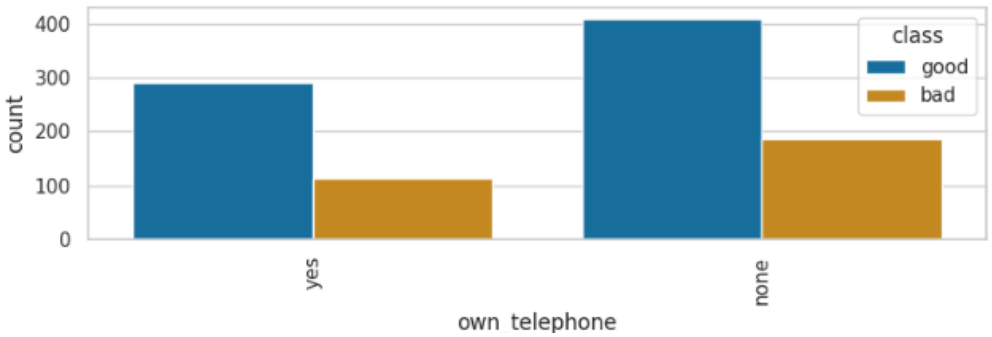
<u>קרדינליות משתנה בין עמודות:</u>

מספר הערכים הייחודיים בקטגוריות משתנה באופן משמעותי מעמודה לעמודה. לדוגמה, בעמודה purpose ישנם 10 ערכים ייחודיים, המייצגים מטרות שונות עבור ההלוואות שנלקחו, כמו רכישת רכב חדש, מימון חינוך, או רכישת רהיטים. לעומת זאת, בעמודות כמו own_telephone, foreign_worker, ו-class יש רק 2 ערכים ייחודיים, מה שמצביע על כך שהן עמודות בינאריות. שונות זו בקרדינליות עשויה לרמז על הבדל בשונות המידע שכל עמודה מספקת, כאשר עמודות עם קרדינליות גבוהה מכילות יותר מידע פוטנציאלי.

	Unique Values	Count of Unique Values
0	[<0, 0<=X<200, no checking, >=200]	4
1	[critical/other existing credit, existing paid...	5
2	[radio/tv, education, furniture/equipment, new...	10
3	[no known savings, <100, 500<=X<1000, >=1000, ...	5
4	[>=7, 1<=X<4, 4<=X<7, unemployed, <1]	5
5	[male single, female div/dep/mar, male div/sep...	4
6	[none, guarantor, co applicant]	3
7	[real estate, life insurance, no known propert...	4
8	[none, bank, stores]	3
9	[own, for free, rent]	3
10	[skilled, unskilled resident, high qualif/self...	4
11	[yes, none]	2
12	[yes, no]	2
13	[good, bad]	2

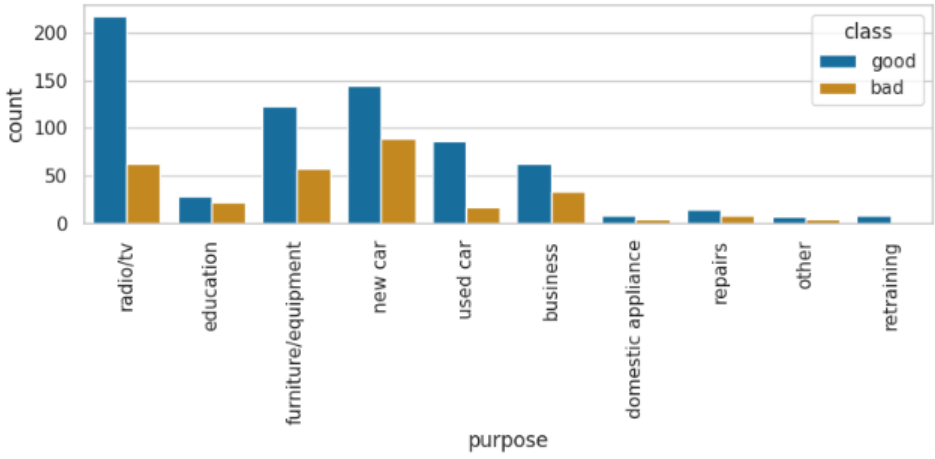
<u>קטגוריות בינאריות:</u>

עמודות כמו own_telephone, foreign_worker, ו-class הן דוגמאות לעמודות בינאריות, כלומר עמודות שיש להן שני ערכים אפשריים בלבד. תכונה זו יכולה להיות רלוונטית במיוחד כשבוחנים תכונות שאולי יש להן השפעה משמעותית על התוצאה הסופית. לדוגמה, נוכל לבדוק האם היותו של העובד זר משפיע על ההסתברות לקבל הלוואה או על דירוג האשראי שלו.



<u>קרדינליות נמוכה לעומת קרדינליות גבוהה:</u>

עמודות עם קרדינליות נמוכה, כמו own_telephone עם 2 ערכים, עשויות להיות קלות לפירוש והבנה. עם זאת, הן עשויות לספק פחות מידע מבחינת המודל, מה שיכול להוביל להשפעה מוגבלת על החיזוי הסופי. לעומת זאת, עמודות עם קרדינליות גבוהה, כמו purpose עם 10 ערכים ייחודיים, עשויות להכיל מידע רב יותר ולהיות בעלות השפעה משמעותית יותר על המודל. עם זאת, עמודות אלו עשויות לדרוש שיטות ניתוח מורכבות יותר כדי להפיק מהן תובנות.

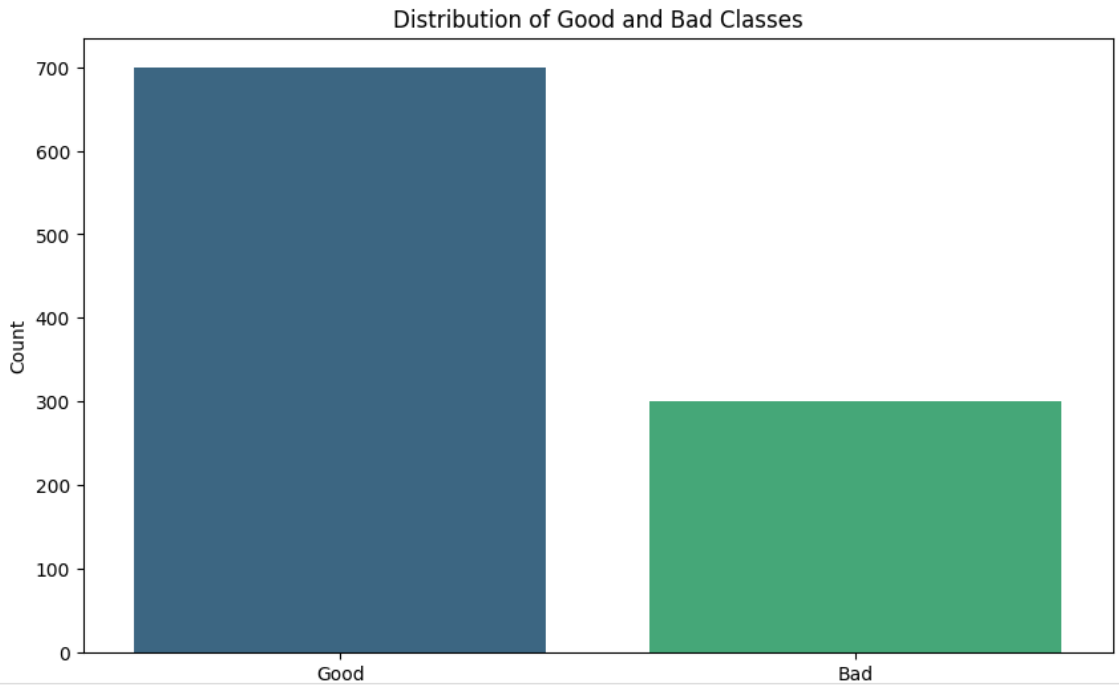


<u>אפשרות לאיחוד קטגוריות:</u>

במהלך הניתוח, ניתן לשקול איחוד של קטגוריות בעמודות מסוימות כדי להקטין את הקרדינליות ולפשט את הניתוח. לדוגמה, בעמודת purpose, אם ישנם ערכים שיכולים להיחשב דומים או זהים מבחינת מטרת האשראי, ניתן לשקול לאחדם לקטגוריה אחת. איחוד זה יכול להקל על הבנת המשתנים ולהפחית את מורכבות המודל.

לאחר מכן ביצענו בדיקה על משתנה המטרה בעזרת הקוד הבא והפלט שמוצג אחריו:

```
plt.figure(figsize=(10, 6))
sns.countplot(x='class', data=df, palette='viridis')
plt.title('Distribution of Good and Bad Classes')
plt.xlabel('Class')
plt.ylabel('Count')
plt.xticks(ticks=[0, 1], labels=['Good', 'Bad'])
plt.show()
class_counts = df['class'].value_counts()
print("Class Distribution:")
print(class_counts)
```

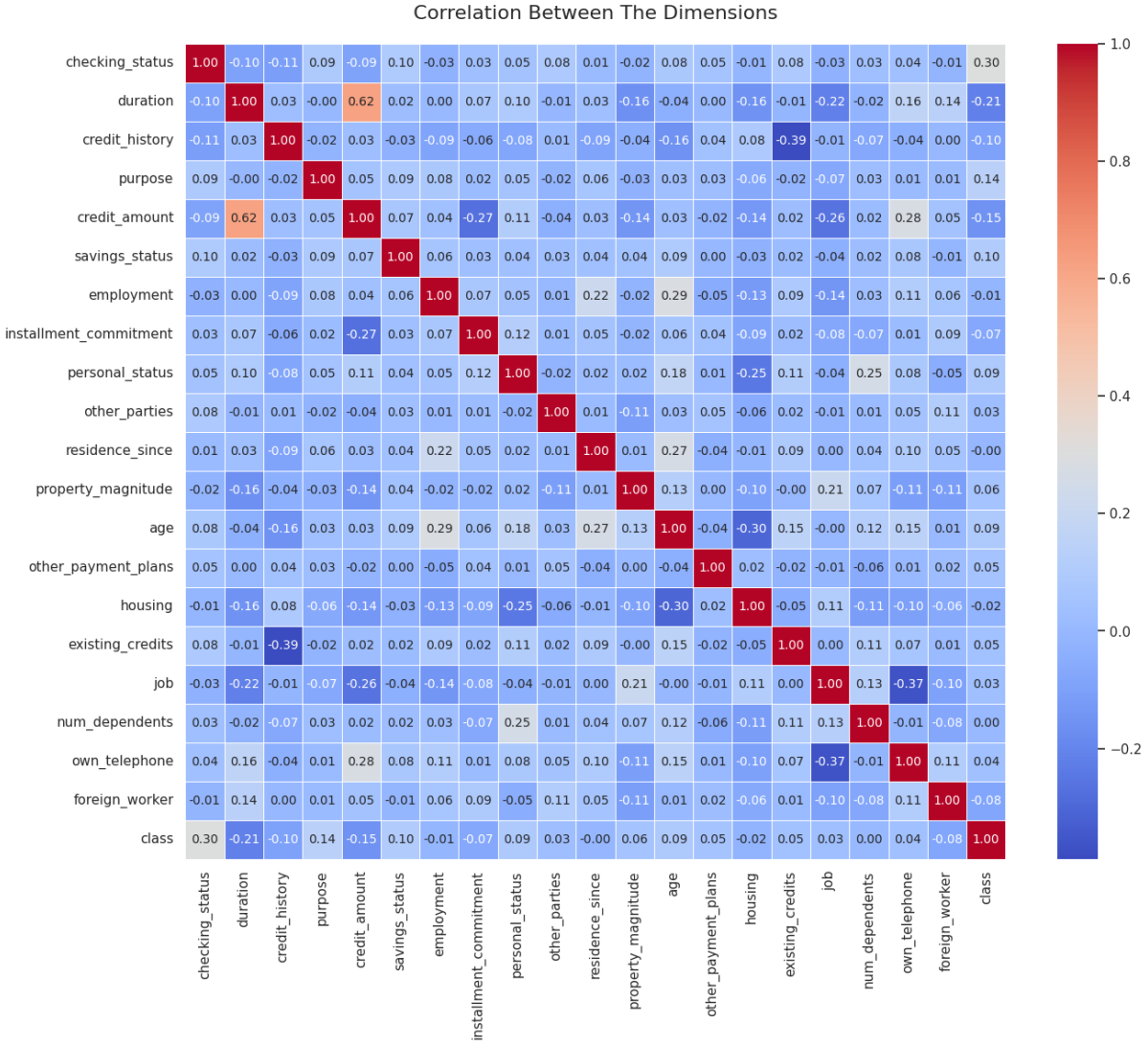


כפי שניתן לראות בתמונה למעלה, חלוקת הנתונים בעמודת המטרה אינה מאוזנת. מתוך 1,000 דוגמאות, כ-70% מהן מסווגות כ "Good"-ורק 30% מסווגות כ ."Bad"-חוסר איזון זה עלול להוות אתגר משמעותי ביכולת החיזוי של המודל, שכן המודל עשוי להעדיף את הסיווג הנפוץ יותר, ובכך להקטין את דיוקו במקרים הפחות נפוצים.

בהתבסס על תובנה זו, כבר בשלב זה הסקנו כי יהיה צורך להשתמש בשיטות מתקדמות להתמודד עם הבעיה, כגון שימוש במודלים כמו Random Forest, אשר יודעים להתמודד בצורה טובה יותר עם נתונים לא מאוזנים. בנוסף, ניתן לשקול שימוש בטכניקות נוספות כגון איזון הנתונים באמצעות Oversampling או Undersampling, או שימוש במטריקות הערכה שמותאמות לבעיות של חוסר איזון כמו ROC-AUC ו-Precision-Recall.

<u>קורלציות בין משתנים:</u>

במהלך שלב הניתוח המקדים, הצגנו פלט שמדגים את הקורלציות בין הממדים השונים בדאטה. פלט זה מספק תובנות חשובות לגבי הקשרים הקיימים בין הפיצ'רים, ומאפשר לנו להבין כיצד תכונות מסוימות משפיעות זו על זו.



בעזרת ה-HEATMAP המציגה את הקורלציה בין כל העמודות בדאטה שלנו, ניתן להסיק מספר תובנות חשובות:

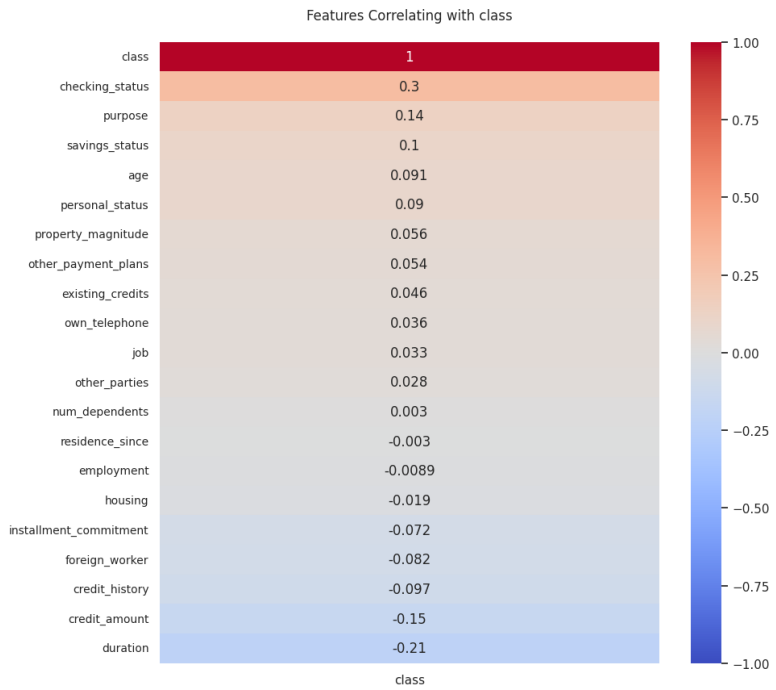
- <u>זיהוי קשרים חזקים בין משתנים:</u> ה-HEATMAP מאפשרת לנו לזהות קשרים חזקים בין משתנים. ערכי קורלציה הקרובים ל-1 מצביעים על קשר חיובי חזק בין שני משתנים, בעוד שערכים הקרובים ל-1- מצביעים על קשר שלילי חזק. לעומת זאת, ערכים הקרובים ל-0 מעידים על היעדר קשר משמעותי בין המשתנים. מתוך מידע זה, ניתן להסיק אילו תכונות עשויות להשפיע יותר על המודל, ואילו תכונות עשויות להוסיף יתירות מסוימת.
- קולינאריות: כאשר אנו מזהים זוגות של משתנים עם קורלציה גבוהה מאוד, יש מקום לחשוד בקולינאריות – מצב שבו משתנים מספקים מידע דומה למודל. במקרים כאלה, ניתן להסיק כי מדובר בייתרות מסוימת, ולכן ניתן לשקול להסיר או לשנות את אחד המשתנים כדי לצמצם את המורכבות ולהגביר את האפקטיביות של המודל.

בסך הכל, ה-HEATMAP שיצרנו סייעה לנו לבחור בתבונה את המשתנים שבהם נשתמש להמשך הניתוח. היא סיפקה לנו הבנה כללית וטובה של הקשרים הקיימים בין הנתונים, מה שמקל על קבלת החלטות מושכלות בנוגע לשימוש במשתנים השונים במודל.

<u>עיבוד מקדים:</u>

Heatmap:

תחילה יצרנו מפה המציגה את הקורלציה בין כל עמודה לבין עמודות הtarget , כך שמשתנים שיש להם קורלציה גבוהה (חיובי או שלילי) עם הtarget יהיו מועמדים עבורנו לשימוש במודל. נסתכל על מפה ונבחן את התובנות:



מפת החום ממחישה את עוצמת הקשר בין כל משתנה לבין משתנה המטרה, מנמוך לגבוה. דבר זה עוזר לנו להבין אילו משתנים משפיעים יותר על התוצאה הסופית, כפי שניתן לראות משתנים עם מתאמים גבוהים כמו checking_status ו-duration מצביעים על כך שהם עשויים להיות חשובים למודל. גם כאן ניתן לראות בפלט על משתנים כמו credit_amount ו-duration המופיעים בתחתית ומשקפים להיותם פחות חשובים למודל. בהתאם, נוכל לבחור את המשתנים החזקים ביותר ולשקול אילו להסיר או לאילו מהם לתת פחות משקל.

יצירת תכונות חדשות:

במהלך תהליך הניתוח, יצרנו מספר תכונות חדשות על בסיס המשתנים הקיימים, במטרה לשפר את ביצועי המודל ולשקף יחסים מורכבים בין המשתנים. להלן סקירה של התכונות החדשות שנוצרו וכיצד הן תורמות לניתוח:

: log_duration

ביצענו טרנספורמציה לוגריתמית על משתני credit_amount ו-duration במטרה להקטין את השפעתם של ערכים חריגים ולהפוך את ההתפלגות שלהם ליותר נורמלית. פעולה זו עשויה לסייע במניעת עיוותים במודל הנובעים מהשפעות של חריגים.

: amount_duration_interaction

יצרנו משתנה חדש המבוסס על האינטרקציה בין שני משתנים קיימים: credit_amount ו-duration. המטרה הייתה לבחון אם השילוב בין גובה האשראי למשך הזמן מהווה גורם חשוב בחיזוי, מכיוון שיש אפשרות שקיימת תלות ביניהם המשפיעה על הסיכון האשראי.

: duration_squared

השתמשנו במשתנים credit_amount ו-duration כדי ליצור תכונות פולינומיאליות, כגון duration_squared המטרה הייתה להעשיר את המודל בתכונות נוספות שיכולו לעזור לו לזהות קשרים לא לינאריים בין המשתנים לבין משתנה המטרה.

:average_credit_amount_by_purpose

חישבנו ממוצע של סכום ההלוואה (credit_amount) לפי מטרת ההלוואה (purpose). תכונה זו משקפת את סכום ההלוואה הממוצע עבור כל מטרה, דבר שיכול להוסיף הקשר חשוב למודל ולעזור בהבנת דפוסי הבקשה השונים.

:purpose_frequency

על מנת להתמודד עם משתנים קטגוריאליים, השתמשנו - frequency encoding למשתנה purpose. בכל קטגוריה הוקצה ערך מספרי בהתאם לתדירות הופעתה, מה שמקל על המודל להתמודד עם משתנים קטגוריאליים.

:lag1_duration

יצרנו משתנה Lag עבור duration, מתוך הנחה שיש קשר סיבתי בין הערכים העוקבים במשתנה זה. תכונה זו עשויה לשפר את דיוק המודל על ידי התחשבות במגמות או תלות על פני זמן.

:rolling_mean_duration, rolling_std_duration

יצרנו משתנים המחשבים את הממוצע (rolling_mean_duration) וסטיית התקן (rolling_std_duration) של המשתנה duration. מטרתנו הייתה לקבל מידע על התנהגות הנתונים בטווחים קצרים יותר, מה שיכול לשפר את הבנת התנהגות המשתנה ולשפר את דיוק המודל.

:Credit Amount Ratio

חישבנו את המחיר הממוצע של כל ההלוואות לפי סיבה מסוימת (purpose), וכן את סטיית התקן עבור כל סיבה. יצרנו משתנה בינארי שמייצג עבור כל רשומה האם סכום ההלוואה המבוקש גבוה משמעותית מהממוצע (גבוה ביותר מחצי סטיית תקן). פעולה זו נעשתה מתוך מחשבה שסכום ההלוואה קשור גם ביחס לסיבת הבקשה, ואם סכום ההלוואה גבוה מאוד יחסית לסיבה, ייתכן שזה מסמן סיכון גבוה יותר.

Monthly Return:

חישבנו את ההחזר החודשי של הלווה על ידי חלוקת סכום ההלוואה המבוקשת במספר החודשים עליהם נפרסת ההלוואה. תכונה זו מאפשרת למודל להעריך את הסיכון האשראי על סמך גובה ההחזר החודשי והיכולת של הלווה לעמוד בתשלומים.

Duration-Age Ratio:

יצרנו משתנה המבטא את היחס בין משך ההלוואה (duration) לגיל הלקוח (age). מטרתנו הייתה לבחון האם רמת הסיכון משתנה בהתאם ליחס זה, מתוך הנחה שלקוחות מבוגרים הנוטלים הלוואות בסכומים נמוכים נחשבים לסיכון נמוך יותר מאשר לקוחות צעירים הנוטלים הלוואות גדולות.

<u>דיסקרטיזציה:</u>

בשלב הניתוח המוקדם, זיהינו שחלק מהפיצ'רים הנומריים שלנו מציגים התפלגויות ייחודיות שאינן נורמליות. הדוגמה המוצגת כאן מתייחסת לפיצ'ר credit_amount, אשר הפגין התפלגות עם זנב ימני ארוך.

לאור המידע הזה, בשלב הראשוני של תהליך הניתוח, החלטנו לחלק את הערכים הנומריים לפלחים (bins) בהתאם להתפלגויות של כל פיצ'ר. החלוקה לפלחים מאפשרת לנו להתייחס לקטגוריות נפרדות של ערכים, מה שמקל על הבנת הקשר בין הפיצ'ר לבין משתנה המטרה ומסייע בהפחתת ההשפעה של ערכים קיצוניים (outliers).

עיצבנו את התהליכים עבור חלק מהפיצ'רים הנומריים שלנו כך שבכל פעם בחנו את התפלגות הערכים ולאחר מכן ביצענו דיסקרטיזציה על מנת לקבץ את הערכים לקטגוריות ברורות. תהליך זה כולל יצירת קטגוריות חדשות לפיצ'רים כמו גיל, משך זמן ההלוואה, גובה ההחזר החודשי, ועוד. כל קטגוריה שהוגדרה מייצגת קבוצה של ערכים מספריים והפכה את המשתנה למשתנה קטגוריאלי, מה שמאפשר למודל להתמודד עם הנתונים בצורה מובנית יותר ולשפר את דיוק התחזיות.

בנוסף, תהליך הדיסקרטיזציה סייע לנו בבניית משתנים מורכבים יותר המבוססים על שילוב של משתנים קיימים. למשל, ביצענו חלוקה של הפיצ'ר duration_age_ratio, המתאר את היחס בין משך ההלוואה לבין גיל הלקוח, לקטגוריות מוגדרות מראש, כך שניתן להבין את האינטראקציה בין שני המשתנים ולהעריך את הסיכון בצורה מדויקת יותר.

```python
def create_age_group(df):
    age_bins = [0, 30, 40, 50, 60, 70, 120]
    age_labels = ['0-30', '31-40', '41-50', '51-60', '61-70', '70+']
    df['age_group'] = pd.cut(df['age'], bins=age_bins, labels=age_labels, include_lowest=True).astype(object)
    df.drop(columns=['age'], inplace=True)

# Creating duration groups
def create_duration_group(df):
    duration_bins = [0, 12, 24, 36, 48, 60, 72]
    duration_labels = ['0-12', '13-24', '25-36', '37-48', '49-60', '61-72']
    df['duration_group'] = pd.cut(df['duration'], bins=duration_bins, labels=duration_labels, include_lowest=True).astype(object)
    df.drop(columns=['duration'], inplace=True)

# Creating duration_age_ratio groups
def create_duration_age_ratio_group(df):
    ratio_bins = [0, 2.5, 5, 7.5, 10, 12.5, 120]
    ratio_labels = ['0-2.5', '2.5-5', '5-7.5', '7.5-10', '10-12.5', '12.5+']
    df['duration_age_ratio_group'] = pd.cut(df['duration_age_ratio'], bins=ratio_bins, labels=ratio_labels, include_lowest=True).astype(object)
    df.drop(columns=['duration_age_ratio'], inplace=True)

# Creating monthly_return groups
def create_monthly_return_group(df):
    return_bins = [0, 50, 100, 150, 200, 250, 50000]
    return_labels = ['0-50', '51-100', '101-150', '151-200', '201-250', '251+']
    df['monthly_return_group'] = pd.cut(df['monthly_return'], bins=return_bins, labels=return_labels, include_lowest=True).astype(object)
    df.drop(columns=['monthly_return'], inplace=True)

# Creating credit_amount groups
def create_credit_amount_group(df):
    amount_bins = [0, 500, 1000, 1500, 2000, 2500, 50000]
    amount_labels = ['0-500', '501-1000', '1001-1500', '1501-2000', '2001-2500', '2501+']
    df['credit_amount_group'] = pd.cut(df['credit_amount'], bins=amount_bins, labels=amount_labels, include_lowest=True).astype(object)
    df.drop(columns=['credit_amount'], inplace=True)
```

Encoding לעמודות קטגוראליות:

הפונקציה label_encode_dataframe נועדה לבצע המרה של ערכים קטגוריאליים בעמודות מסוג object לערכים מספריים, על ידי שימוש בטכניקת Label Encoding. מדובר בטכניקה חיונית כאשר אנחנו עובדים עם משתנים קטגוריאליים במודלים של למידת מכונה, שכן מודלים רבים אינם מסוגלים להתמודד ישירות עם ערכים לא מספריים.

הפונקציה מבצעת המרה של עמודות קטגוריאליות ב-DataFrame לערכים מספריים באמצעות טכניקת Label Encoding. היא מתחילה בהעתקת ה-DataFrame המקורי כדי לשמור על נתוני המקור, ולאחר מכן מזהה את כל העמודות שהן מסוג object (קטגוריאליות). עבור כל עמודה מזוהה, הפונקציה יוצרת אובייקט LabelEncoder שממיר את הערכים הקטגוריאליים לערכים מספריים ייחודיים. בנוסף להמרה, הפונקציה שומרת את ה-Label Encoder במילון כך שניתן יהיה להשתמש בו בעתיד לצורך שחזור הערכים או לעיבוד נתוני בדיקה. לבסוף, הפונקציה מחזירה את ה-DataFrame שהומר יחד עם המילון המכיל את ה- Label Encoders, מה שמאפשר לנו להמשיך בתהליך עיבוד הנתונים בצורה יעילה.

שימוש בפונקציה זו מאפשר לנו להפוך את המשתנים הקטגוריאליים למשתנים מספריים בצורה אוטומטית וקלה לשימוש, מה שמאפשר לנו לשלב אותם ישירות במודלים של למידת מכונה. הפונקציה משמשת כחלק חשוב מתהליך הטרנספורמציה וההכנה של הנתונים, ומבטיחה שהמודלים שלנו יוכלו להתמודד בצורה מיטבית עם כל סוגי המשתנים.

חלוקה ל-train ו-test ואיזון הנתונים:

במסגרת תהליך פיתוח המודל על פי מתודולוגיית CRISP-DM, לאחר שלב ההכנה והעיבוד של הנתונים, עברנו לשלב בניית המודלים שבו ביצענו את חלוקת הנתונים לסטים של אימון ובדיקה, ולאחר מכן יישמנו את טכניקת SMOTE להתמודדות עם בעיית חוסר האיזון בנתונים.

תחילה, חילקנו את הנתונים לסטי אימון ובדיקה. המטרה בחלוקה זו היא להפריד את הנתונים כך שנוכל לאמן את המודל על חלק מהנתונים ולבחון את ביצועיו על חלק אחר שלא נחשף אליו במהלך האימון. לשם כך, השתמשנו בפונקציה train_test_split, כאשר 80% מהנתונים הוקצו לסט האימון ו-20% לסט הבדיקה.

```
# Splitting data into training and testing sets
y = encoded_df['class']
X = encoded_df.drop(['class'], axis=1)
std_scaler = StandardScaler()
X_scaled = std_scaler.fit_transform(X)
X = pd.DataFrame(X_scaled, columns=X.columns)
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

בתהליך החלוקה של הנתונים לסטי אימון ובדיקה, בוצעה גם סטנדרטיזציה על המשתנים באמצעות השימוש ב-StandardScaler. סטנדרטיזציה זו מבטיחה שכל המשתנים יהיו באותה סקאלה, בדרך כלל על ידי קביעת ממוצע השווה לאפס וסטיית תקן השווה לאחד. פעולה זו חשובה במיוחד במודלים של למידת מכונה, מכיוון שמשתנים עם סקאלות שונות יכולים להשפיע באופן לא פרופורציונלי על ביצועי המודל. באמצעות הסטנדרטיזציה, אנו מבטיחים שהמודל יתמקד במבנה הפנימי של הנתונים – כלומר, בקשרים בין המשתנים השונים ובדפוסים שמובילים לתוצאה ולא בהבדלים המספריים בסקאלה של המשתנים עצמם. זהו שלב קריטי בתהליך ה Modeling-במסגרת CRISP-DM, שכן הוא מאפשר בניית מודלים יציבים ומדויקים יותר, המגיבים היטב לשינויים ולהתפלגויות השונות של הנתונים.

לאחר חלוקת הנתונים, יישמנו את טכניקת SMOTE (Synthetic Minority Over-sampling Technique) על סט האימון. טכניקת SMOTE נועדה להתמודד עם חוסר איזון בין הקטגוריות במשתנה המטרה (class), כאשר אחת הקטגוריות מופיעה הרבה פחות מאשר הקטגוריה האחרת כפי שהצגנו בעיבוד המקדים. SMOTE יוצר דגימות סינתטיות חדשות עבור הקטגוריה המיעוטית על ידי שילוב של דגימות קיימות. בכך, הוא מגדיל את ייצוג הקטגוריה המיעוטית בסט האימון, ומשפר את יכולתו של המודל להבחין בין הקטגוריות השונות, במיוחד בקטגוריה המיעוטית.

```
# Apply SMOTE to the training data
smote = SMOTE(random_state=42)
x_train_smote, y_train_smote = smote.fit_resample(x_train, y_train)
```

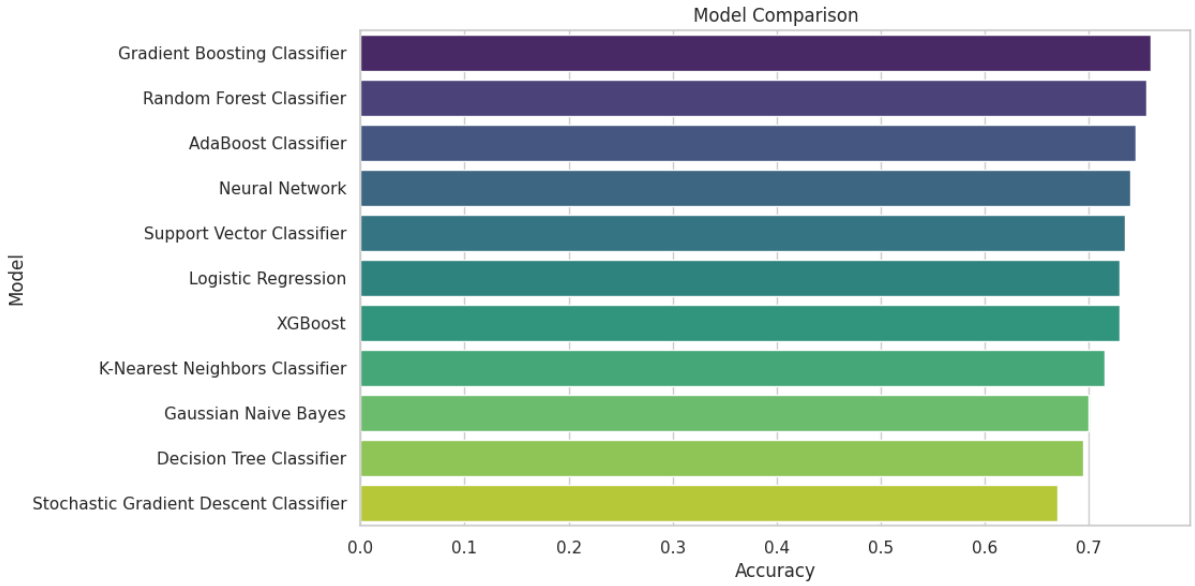
תהליך זה מהווה חלק מרכזי בשלב ה-Modeling של CRISP-DM, ומאפשר לנו לבנות מודל חזק ועמיד יותר, שמסוגל להתמודד עם האתגרים של חוסר איזון בנתונים ולספק תחזיות מדויקות יותר. השילוב של חלוקת הנתונים ויישום SMOTE מבטיח שהמודל לא רק ילמד בצורה טובה יותר את הדפוסים הקיימים, אלא גם יוכל להכליל את התחזיות שלו בצורה אפקטיבית יותר על נתונים חדשים.

מודלים ובחירה:

בשלב ההתחלתי רצינו להריץ מודלים טריוויאלים על מנת שנוכל לקבל בייסליין סביר שנוכל לעבוד מולו (כאשר קיים גם הבייסליין של המחברות בקאגל שמכילות הרבה יותר פיצ'רים מורכבים לכן בשלב זה אינן היו לנו רלוונטית)

המודלים הראשונים שיצרנו ללא Feature engineering הראו תוצאות מבטיחות:

8	Gradient Boosting Classifier	0.760	0.808293
0	Random Forest Classifier	0.755	1.065662
10	AdaBoost Classifier	0.745	0.572176
3	Neural Network	0.740	2.834652
6	Support Vector Classifier	0.735	0.122412
2	Logistic Regression	0.730	0.043907
4	XGBoost	0.730	1.250646
9	K-Nearest Neighbors Classifier	0.715	0.050652
7	Gaussian Naive Bayes	0.700	0.024197
1	Decision Tree Classifier	0.695	0.032465
5	Stochastic Gradient Descent Classifier	0.670	0.052071



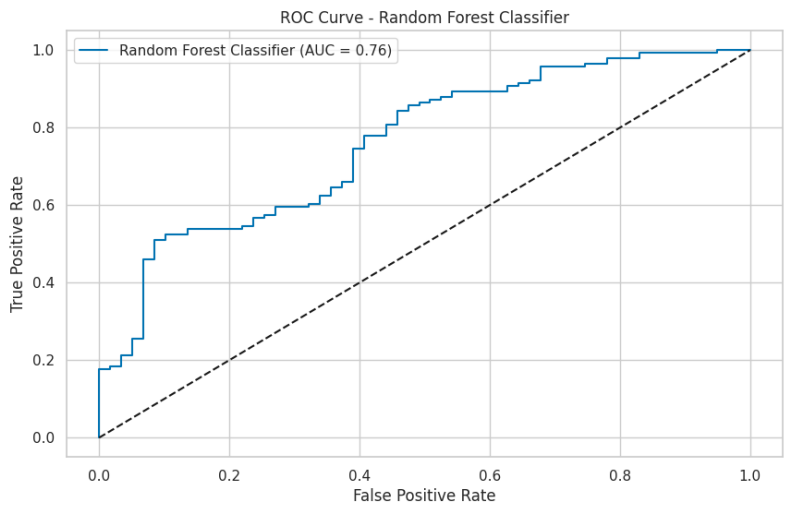
כלומר ראינו ביצועים מרשימים כבר מההתחלה אך המחברות בקאגל הראו בייסליין של 80% דיוק ולכן רצינו לראות האם מה שיקדם אותנו למטרה הזאת תיהיה יצירת פיצ'רים חדשים או בעזרת חיפוש היפרפרמטרים טובים יותר עבור המודלים. בגלל שמודל הRandom forest והBoosting הציגו תוצאות דומות בחרנו להתקדם עם Random forest כיוון ששיטות בוסטינג הם יותר מדויקות ומתקנות טעויות שהמודלים הקודמים מראים, כלומר הבייסליין שם יהיה יותר מדויק כיוון שכל מודל הבא בBoosting יתקן את השגיאות של הקודם ועל כן העדפנו להשתמש בשיטת Ensamble לאחר Fine tune .

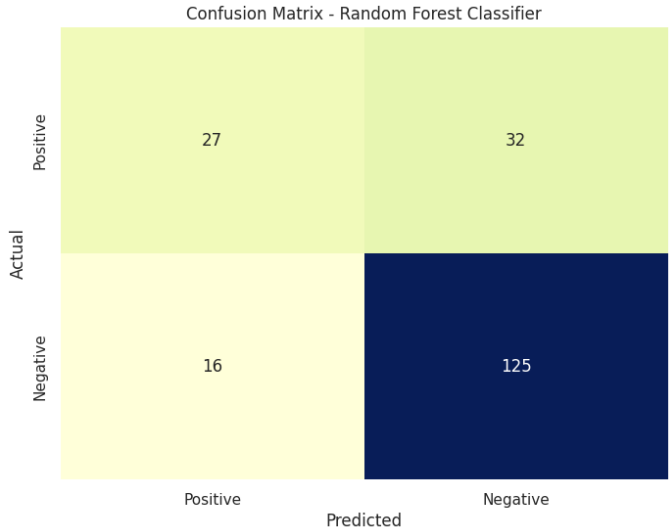
לאחר שביצענו Fine tune למודל בעזרת ההיפר פרמטרים המאוד מקיפים הבאים :

```
params = {'n_estimators': [10, 100,500],
                'max_features': ['auto', 'sqrt'],
                'max_depth': [3,4,5,6,8,10,12,15],
                'min_samples_split': [3,4,5,6,8,10,12,15],
                'min_samples_leaf': [1,2,3,4],
                'bootstrap': [True, False]
}
```

קיבלנו את התוצאות הבאות:

Model	Accuracy	Precision	Recall	F-Score	ROC AUC	Training Time
Random Forest Classifier	0.76	0.796178	0.886525	0.838926	0.760067	2.866589





אמנם התוצאות האלו מבטיחות כיוון ש 76% דיוק במטלה כזאת זה לא רע בכלל, בניתוח הנתונים שהתקבלו ראינו כי הPrecision שואף ל80% שזה לא אידאלי ונרצה לשפר את הPrecision עוד כיוון שבמטלה מסוג אנחנו נעדיף כמובן Precision יותר גבוה כדי בדיוק להימנע מהאפשרות של בנקים אשר פועלים בצורה לא

חכמה. אך בכל מקרה קיבלנו את הממוצע של הPrecision וRecall באזור ה0.83 שזה שלא תוצאה רעה בכלל ומראה על איזון טוב בין הPrecision לRecall . נוסף על כך ראינו כי הRoc AUC הינו באזור ה0.76 שזה כבר נורה אדומה במטלה מהסוג הזה, בעצם זה אומר האם המודל מפריד בצורה טובה בין הקלאסים. אומנם 0.76 זה לא תוצאה רעה מידי אך במטלה מסוג זה הROC AUC הוא כנראה הכי קריטי כיוון שאנחנו בדיוק מנסים להימנע מבעיות של קריסת בנק עקב הלוואות אגריסיביות או הפסד שלו עקב מחסור בהלוואות וROC AUC זאת לדעתי המטריקה הכי חשובה בפרוייקט הזה עבור הביצועים של המודל, כיוון שאם הוא לא יודע להבדיל בצורה מספקת בין הקלאסים אז אין משמעות לפרוייקט ולעבודה וזה לא תורם בשום צורה לאישור הלוואות. בנוסף לזה בConfusion ניתן לראות חוסר איזון מסויים בין הקלאסים ועל כן זה יוצר בעיה קטנה אך נתייחס לזה כמצב מציאותי ולא ניתן לו Seed חדש כדי לטפל בזה.

לאחר כל התובנות הללו השלב הבא שמתבקש זה בעצם לחזור לPreprocessing וFeature engineering לפי CRISP-DM ולאחר כל התובנות מהעבודה וכ10 איטרציות של CRISP-DM כל מה שמוצג בחלק של הPreprocessing זה רלוונטי לשלב הסופי, וכך גם המודלים הבאים שנרחיב עליהם.

מודלים והיפר פרמטרים:

עבור כל אחד מהמודלים הבאים פרט לEnsemble neuralnetwork בסוף ביצענו Cross validation של 5 עם Gridsearch בעזרת הפונקציה Randomized search .

Logistic regression :

את החיפוש ביצענו על הפרמטרים:

```
param_grid = {
    'penalty': ['l1', 'l2', 'elasticnet', 'none'],
    'C': np.logspace(-4, 4, 20),
    'solver': ['lbfgs', 'liblinear', 'saga']
}

log_reg = LogisticRegression(max_iter=1000)
log_reg_cv = RandomizedSearchCV(log_reg, param_grid, n_iter=50, cv=5, random_state=42, n_jobs=-1, scoring='accuracy')
```

עם הגבלת המודל על 1000 איטרציות כדי להוריד את הזמן חיפוש (בניסיון הראשון הגבלנו ל5000 ו1000 ואף אחד מהם לא חרג)

בחיפוש הגדרנו כי נחפש עבור כל הפרמטרים ונבצע 50 איטרציות (כלומר הכמות פרמטרים אשר זה דוגם, נובע מהכמות הגבוה בLogspace). הגדרנו CV של 5 כלומר 5 Fold cross validation (20% שמור לולידציה ו80% לאימון מתוך הTraining set) עם Seed 42 על מנת שנוכל לשחזר את התוצאות. ההיפרפרמטרים נבחרו לפי מטריקת דיוק (Accuracy) כיוון שבמודלי קלסיפיקציה ובמיוחד בתחום המשין לרנינג מאוד מומלץ להשתמש בזה.

לאחר מכן מצאנו כי הפרמטרים הטובים ביותר עבור המודל הינם:

```
▾ LogisticRegression
LogisticRegression(C=0.615848211066026, max_iter=1000, penalty='l1',
                   solver='liblinear')
```

כלומר הפרמטרים הטובים ביותר שנבחרו הינם: C=0.6 כלומר לא ניתן משמעות חזקה מידי לרגולריזציה (ככל שהמספר יותר קרוב ל0 ככה הרגולריזציה משפיעה יותר). נשתמש ברגולריזציה מסוג Lasso

נוסף לכך גם נבחר אלגוריתם אופטימיזציה מסוג Liblinear אשר מעולה לדאטהסטים קטנים וקלאסים בינאריים (כאשר Saga טוב למולטיקלאס אך השארתי אותו כאן על מנת לבדוק, אותו דבר נכון עבור Lbfgs)

XGBoost:

כמו שציינתי מקודם גם רציתי לבדוק את ההשפעה של מודלי בוסטינג על הדאטה והמטלה ולכן בחרתי להשתמש בXGBoost למטלה זאת.

את חיפוש הפרמטרים ביצעתי על הפרמטרים הבאים:

```
param_grid = {
    'n_estimators': [100, 200, 300, 400, 500],
    'learning_rate': [0.001, 0.01, 0.1, 0.2, 0.3],
    'max_depth': [3, 4, 5, 6, 7, 8],
    'min_child_weight': [1, 2, 3, 4, 5],
    'subsample': [0.6, 0.7, 0.8, 0.9, 1.0],
    'colsample_bytree': [0.6, 0.7, 0.8, 0.9, 1.0],
    'gamma': [0, 0.1, 0.2, 0.3, 0.4, 0.5]
}

xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss')
xgb_cv = RandomizedSearchCV(xgb, param_distributions=param_grid, n_iter=50, cv=5, random_state=42, n_jobs=-1, scoring='accuracy')
xgb_cv.fit(x_train, y_train)
```

כאשר N מייצג את כמות הBoosting אשר נבצע (כלומר סיבובי הבוסטינג, הLearning rate מייצג כמה מהר הוא לומד על המידע את השיפורים לבצע לאיטרציה הבאה. הMax depth מייצג את עומק העץ עבור כל "לומד", Min child weight מייצג את סכום המשקל המינמלי שחייב לתת לכל ילד (כלומר כל לומד שיוצא מהעץ הבא. Subsample מייצג את היחס דגימה עבור כל סטי הדוגמאות, וכנל גם Colsample אך זה עבור כל עץ שנבנה. ולבסוף Gamma מייצג את הLoss המינמלי שכל עץ במודל יחליט לפיו האם לבצע עוד פיצול בעלה.

הCross validation בוצע בצורה דומה לLogistic regression עם -1 ג'ובים כדי לתת את כל המשאבי CPU שקיימים בColab למודל. בנוסף המטריקה עבור המודל עצמו הינה Logloss .

והתקבל כי אלה הפרמטרים האידאליים בחיפוש שלנו:

```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.9, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric='logloss',
              feature_types=None, gamma=0.5, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=0.01, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=6,
              max_leaves=None, min_child_weight=5, missing=nan,
              monotone_constraints=None, multi_strategy=None, n_estimators=500,
              n_jobs=None, num_parallel_tree=None, random_state=None, ...)
```

כולל כאלה שלא הגדרנו.

Random forest:

מודל זה הינו מודל Ensemble ומאוד מעניין אותנו כיוון שראינו ביצועים טובים שלו בהתחלה וניסינו כבר XGBoost כלומר מודל Boosting ועל כן ננסה גם עכשיו מודל מסוג Ensemble על מנת לבדוק את הכוח שלהם על מטלה מסוג זה לפניי שנתקדם לשיטות יותר מורכבות שאולי ישלבו את אחת מהשיטות האלה.

את החיפוש עשינו על ההיפר פרמטרים הבאים:

```
params = {'n_estimators': [10, 100,500],
                'max_features': ['auto', 'sqrt'],
                'max_depth': [3,4,5,6,8,10,12,15],
                'min_samples_split': [3,4,5,6,8,10,12,15],
                'min_samples_leaf': [1,2,3,4],
                'bootstrap': [True, False]
}
```

כאשר n_estimator מגדיר את כמות העצים ב"יער", והוא נע בין כמות קטנה של עצים לכמות ענקית.
Max_Features זה בעצם כמות הפיצ'רים שהוא יצטרך להסתכל עליהם כאשר הוא מחליט האם לבצע פיצול בעץ בעצם הוא מחליט על כמו פיצ'רים או שלוקח את השורש של כמות הפיצ'רים שיש לנו. Max_depth מגדיר את העומק המקסימלי של כל עץ כאשר רצינו לתחום את זה כדי לא להביא למצב של Overfit על הפיצ'רים. Min_samples_split זה הכמות המינמלית של הרשומות שקיימות כרגע בצומת החלטה כדי שנוכל לפצל, כלומר אם הגיעו יותר מ3 רשומות לעלה אז המודל יכול לשקול האם לפצל את העלה.
Min_samples_leaf זה כמות הרשומות המינמלית של עלה. כלומר זה גם מעין עצירה כזאת כדי להימנע מOverfit , בנוסף לכך אם הגדרנו שהכמות רשומות בשביל להחליט האם לעשות פיצול היא 3 בעלה אבל הגדרנו פה גם 3 אז המודל יתעלם מכך ולא יבצע פיצול באותו העלה. Boostrap זה האם להשתמש בכמות מהדאטה או להשתמש בכולו כל פעם מחדש.

:SVC

יש לציין שאומנם בהתחלה הגדרנו את זה כSVM אך זהו פשוט שם כולל לכל שיטות ה Support vector machine ובתוכו מכיל המון שיטות כגון ל Linear svc, stochastic gradient svc ועוד. בחרנו ספציפית בזה בגלל שהבעיה שלנו הינה קליספיקציה בינארית והשיטה הספציפית הזאת בודקת את הנקודות של הפיצ'רים אחד מול השני ולא אחד מול כולם.

מודל זה אמור להיות מאוד חזק במימד של הפיצ'רים להפרדה בין קלאסים שונים ואם הגדרה נכונה ומספיק פיצ'רים נומרים הוא אמור לבצע עבודה מעולה בהשוואה לכל המודלים הקודמים (לא כולל הXGboost שזו שיטת Boosting מורכבת) ולכן רצינו לבדוק אותו

נבצע את החיפוש על הפי'צרים הבאים:

```
param_grid = {
    'C': [0.1, 1, 10, 100, 1000],
    'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
    'kernel': ['linear', 'poly', 'rbf', 'sigmoid']
}

svm = SVC(probability=True)
svm_cv = RandomizedSearchCV(svm, param_distributions=param_grid, n_iter=50, cv=5, random_state=42, n_jobs=-1, scoring='accuracy')
svm_cv.fit(x_train, y_train)
```

כלומר הגדרנו כי פרמטר רגולריזציה C שיחפש עליו, יש לציין שזה מסוג Ridge . נוסף על כך הגדרנו לו להשתמש בכל האלגוריתמים שקיימים עבור פתרון הבעיה הזאת כאשר כל אחד מהשיטות שונה בסוג המשוואה שהיא מרכיבה ואמינה לשם שלה כלומר Linear יוצר משוואה לינארית לחישוב וRBF לדוגמה יוצר משוואה רדיאלית אשר לוקח בחשבון את הזוויות. בעיקרון אלגוריתם לינארי הוא טוב בשביל המרחב Input שלנו ונראה זאת גם בהמשך, כאשר האלגוריתמים האחרים יותר טובים לפתרון בעיות יותר מסובכות (ובעיקרון Sigmoid פשוט לא יעיל חישובית בשום צורה ולכן נמנע ממנו כמה שיותר) וGamma הינו רלוונטי רק למודלים שהם לא Linear , אבל זה בעיקרון מגדיר את המקדם גאמא עבור המשוואת כפי שניתן לראות כאן:

Linear Kernel: $K(X,Y) = X^T Y$

Polynomial kernel: $K(X,Y) = (\gamma \cdot X^T Y + r)^d, \gamma > 0$

Radial basis function (RBF) Kernel: $K(X,Y) = \exp(\|X-Y\|^2/2\sigma^2)$ which in simple form can be written as $\exp(-\gamma \cdot \|X-Y\|^2), \gamma > 0$

Sigmoid Kernel: $K(X,Y) = \tanh(\gamma \cdot X^T Y + r)$ which is similar to the sigmoid function in logistic regression.

לבסוף כצפוי ולפי מה שהוסבר התקבל כי הפרמטרים האידאליים הם:

```
SVC
SVC(C=1000, gamma=1, kernel='linear', probability=True)
```

הגאמא לא רלוונטי כפי שניתן לראות במשוואה עם פרמטר רגולריזציה של 1000 כלומר רגולריזציה מאוד קשוחה על המודל.

Multi layer perceptron:

מודל מאוד טריוויאלי והכי בסיסי של רשתות נוירונים המאפשר עבודה קלה וגמישות עם הקוד. זה בעצם מודל של פרספטרון שמגיע מוכן כיביכול ופשוט צריך להגדיר לו את הפרמטרים ונקבל את הרשת ניורנים הכי בסיסית שקיימת.

חיפשנו את הפרמטרים עבור:

```
param_grid = {
    'hidden_layer_sizes': [(50,), (100,), (50, 50)],
    'activation': ['tanh', 'relu'],
    'solver': ['sgd', 'adam'],
    'alpha': [0.0001, 0.05],
    'learning_rate': ['constant','adaptive'],
    'learning_rate_init': [0.001, 0.01, 0.1]
}
mlp = MLPClassifier(max_iter=1000)
mlp_cv = RandomizedSearchCV(mlp, param_distributions=param_grid, n_iter=50, cv=5, random_state=42, n_jobs=-1, scoring='accuracy')
mlp_cv.fit(x_train, y_train)
```

כאשר המודל משתמש בשתיי שכבות לכל היותר של 50 או 100 Perceptrons בשכבה הראשונה או 50 בראשונה ו50 בשניה, בעצם נוצר כאן מודל מאוד פשוט ומנוון אך אמור להיות אפקטיבי למטלה כזאת. אמנם נראה כי זה הרבה עבור שכבה בשביל 20 פיצ'רים אך יש לציין שיצרתי עוד פיצ'רים בשלב זה ולפי ניתוח התוצאות לא נראה כי המודל עושה Overfit ואפילו נוטה לכיוון הUnderfit . נראה זאת בשלב התוצאות.

יותר מכך הגדרנו שכבת הפעלה של Tanh או Relu עבור המטלה. אלגוריתם אופטימזציה של Stochastic gradient decent או Adam . עם המקדם Alpha עבור הרגולריזציה שלא הבאתי להם יותר מידי פרמטרים כיוון שגם ככה נראה לי כי המודל אינו עושה Overfit לפי התוצאות שקיבלתי. בנוסף לכך הגדרתי לו שהמקדם למידה יהיה קבוע כל עוד הLoss יורד או ישתנה כל עוד הLoss לא יורד או ישאר קבוע בין האיטרציות השונות.

הפרמטרים האידאליי שהתקבלו הינם:

```
{'solver': 'sgd', 'learning_rate_init': 0.001, 'learning_rate': 'adaptive', 'hidden_layer_sizes': (100,), 'alpha': 0.0001, 'activation': 'tanh'}
```

שאכן ניתן לראות שזה מפתיע מאוד כי התרגלנו לכך שאופטמייזר Adam תמיד יציג את התוצאות הטובות ביותר וגם נראה שהמודל העדיף לעבוד עם שכבה אחת של 100 ניורנים מאשר לפצל ל2 שכבות. נוסף לכך עוד משהו מאוד מפתיע שבזה עוד לא נתקלתי בעבודות זה העדפה להשתמש באקטיבצית Tanh ולא Relu . אני אישית מאמין שכל זה נובע מכך שזה מודל מאוד מנוון ואם היינו מוסיפים עוד הרבה שכבות קטנות יותר שמתרחבות ואז מצטמצמות כנראה שהיה העדפה לRelu וAdam עם כמות ניורנים קטנים בשכבות הראשונות ויותר בשכבות האמצעיות.

כמצופה וכנאמר למעלה המודל כנראה עושה מעין Underfit שנראה גם בתוצאות אך גם ניתן לראות שהוא בחר ברגולריזציה הנמוכה יותר מה שמראה שאין פה שום קשר לOverfit.

Ensemble FFNN:

מודל זה הוא מודל משמעותית יותר מורכב ממודל הMLP . הוא בעצם משתמש ברשתות נוירונים אך לאורך העבודה ראינו גם ששיטות Ensamble עובדות טוב מאוד למטלה מהסוג הזה אך בחרנו גם לעשות Ensamble עבור רשתות ניורנים. בעצם התהליך היה לעשות מעין Gridsearch על הפרמטרים האפשריים ולראות איזה מודלים מביאים ביצועים יותר טובים (למרות שבאנסמבל זאת לא באמת השיטה כי אולי הם מסתמכים על אותם הפיצ'רים אך בפן התוצאות זה עבד מעולה)

כל רשת בודדת מורכבת מהשכבות הבאות:

שכבה אחת בין המימד פיצ'רים ל128 ניורנים, Batchnorm כדי לעזור להתכנסות של המודל, והפעלה מסוג Relu והוספת Dropout להימנע מOverfit בגלל מורכבות המודל.

התהליך דומה עבור כל שאר השכבות כאשר יש שכבה בין ה128 ניורנים הקודמים ל64 ואז ל32 ולבסוף בין 32 ל1 שזה בעצם הקלסיפקציה הבינארית. בשכבה האחרונה אני משתמש בSigmoid.

עבור אותה הרשת חיפשנו את ההיפרפרמטרים עם הביצועים הכי טובים מתוך ההיפרפרמטרים הבאים:

```
param_grid = {
    'dropout_rate': [0.3, 0.5, 0.7],
    'lr': [0.001, 0.01, 0.1],
    'epochs': [50, 100],
    'optimizer': [optim.Adam, optim.SGD]
}
```

והתקבל כי ה5 מודלים עם הביצועים הכי טובים היו (כאשר החישוב של הAccuracy בוצע כבר על ה20% טסט עצמו) :

```
Params: {'dropout_rate': 0.3, 'epochs': 50, 'lr': 0.1, 'optimizer':
<class 'torch.optim.adam.Adam'>}, Accuracy: 0.83
Params: {'dropout_rate': 0.3, 'epochs': 100, 'lr': 0.01, 'optimizer':
<class 'torch.optim.sgd.SGD'>}, Accuracy: 0.81
Params: {'dropout_rate': 0.3, 'epochs': 100, 'lr': 0.1, 'optimizer':
<class 'torch.optim.sgd.SGD'>}, Accuracy: 0.815
Params: {'dropout_rate': 0.7, 'epochs': 100, 'lr': 0.01, 'optimizer':
<class 'torch.optim.adam.Adam'>}, Accuracy: 0.795
Params: {'dropout_rate': 0.3, 'epochs': 50, 'lr': 0.01, 'optimizer':
<class 'torch.optim.sgd.SGD'>}, Accuracy: 0.79
```

מתוך כל המודלים אשר נראה ככה:

```
Params: {'dropout_rate': 0.3, 'epochs': 50, 'lr': 0.001, 'optimizer': <class 'torch.optim.adam.Adam'>}, Accuracy: 0.78
Params: {'dropout_rate': 0.3, 'epochs': 50, 'lr': 0.001, 'optimizer': <class 'torch.optim.sgd.SGD'>}, Accuracy: 0.775
Params: {'dropout_rate': 0.3, 'epochs': 50, 'lr': 0.01, 'optimizer': <class 'torch.optim.adam.Adam'>}, Accuracy: 0.725
Params: {'dropout_rate': 0.3, 'epochs': 50, 'lr': 0.01, 'optimizer': <class 'torch.optim.sgd.SGD'>}, Accuracy: 0.79
Params: {'dropout_rate': 0.3, 'epochs': 50, 'lr': 0.1, 'optimizer': <class 'torch.optim.adam.Adam'>}, Accuracy: 0.83
Params: {'dropout_rate': 0.3, 'epochs': 50, 'lr': 0.1, 'optimizer': <class 'torch.optim.sgd.SGD'>}, Accuracy: 0.775
Params: {'dropout_rate': 0.3, 'epochs': 100, 'lr': 0.001, 'optimizer': <class 'torch.optim.adam.Adam'>}, Accuracy: 0.745
Params: {'dropout_rate': 0.3, 'epochs': 100, 'lr': 0.001, 'optimizer': <class 'torch.optim.sgd.SGD'>}, Accuracy: 0.775
Params: {'dropout_rate': 0.3, 'epochs': 100, 'lr': 0.01, 'optimizer': <class 'torch.optim.adam.Adam'>}, Accuracy: 0.75
Params: {'dropout_rate': 0.3, 'epochs': 100, 'lr': 0.01, 'optimizer': <class 'torch.optim.sgd.SGD'>}, Accuracy: 0.81
Params: {'dropout_rate': 0.3, 'epochs': 100, 'lr': 0.1, 'optimizer': <class 'torch.optim.adam.Adam'>}, Accuracy: 0.755
Params: {'dropout_rate': 0.3, 'epochs': 100, 'lr': 0.1, 'optimizer': <class 'torch.optim.sgd.SGD'>}, Accuracy: 0.815
Params: {'dropout_rate': 0.5, 'epochs': 50, 'lr': 0.001, 'optimizer': <class 'torch.optim.adam.Adam'>}, Accuracy: 0.765
Params: {'dropout_rate': 0.5, 'epochs': 50, 'lr': 0.001, 'optimizer': <class 'torch.optim.sgd.SGD'>}, Accuracy: 0.755
Params: {'dropout_rate': 0.5, 'epochs': 50, 'lr': 0.01, 'optimizer': <class 'torch.optim.adam.Adam'>}, Accuracy: 0.77
Params: {'dropout_rate': 0.5, 'epochs': 50, 'lr': 0.01, 'optimizer': <class 'torch.optim.sgd.SGD'>}, Accuracy: 0.775
Params: {'dropout_rate': 0.5, 'epochs': 50, 'lr': 0.1, 'optimizer': <class 'torch.optim.adam.Adam'>}, Accuracy: 0.775
Params: {'dropout_rate': 0.5, 'epochs': 50, 'lr': 0.1, 'optimizer': <class 'torch.optim.sgd.SGD'>}, Accuracy: 0.76
Params: {'dropout_rate': 0.5, 'epochs': 100, 'lr': 0.001, 'optimizer': <class 'torch.optim.adam.Adam'>}, Accuracy: 0.765
Params: {'dropout_rate': 0.5, 'epochs': 100, 'lr': 0.001, 'optimizer': <class 'torch.optim.sgd.SGD'>}, Accuracy: 0.775
Params: {'dropout_rate': 0.5, 'epochs': 100, 'lr': 0.01, 'optimizer': <class 'torch.optim.adam.Adam'>}, Accuracy: 0.78
Params: {'dropout_rate': 0.5, 'epochs': 100, 'lr': 0.01, 'optimizer': <class 'torch.optim.sgd.SGD'>}, Accuracy: 0.79
Params: {'dropout_rate': 0.5, 'epochs': 100, 'lr': 0.1, 'optimizer': <class 'torch.optim.adam.Adam'>}, Accuracy: 0.785
Params: {'dropout_rate': 0.5, 'epochs': 100, 'lr': 0.1, 'optimizer': <class 'torch.optim.sgd.SGD'>}, Accuracy: 0.765
Params: {'dropout_rate': 0.7, 'epochs': 50, 'lr': 0.001, 'optimizer': <class 'torch.optim.adam.Adam'>}, Accuracy: 0.78
Params: {'dropout_rate': 0.7, 'epochs': 50, 'lr': 0.001, 'optimizer': <class 'torch.optim.sgd.SGD'>}, Accuracy: 0.755
Params: {'dropout_rate': 0.7, 'epochs': 50, 'lr': 0.01, 'optimizer': <class 'torch.optim.adam.Adam'>}, Accuracy: 0.79
Params: {'dropout_rate': 0.7, 'epochs': 50, 'lr': 0.01, 'optimizer': <class 'torch.optim.sgd.SGD'>}, Accuracy: 0.77
Params: {'dropout_rate': 0.7, 'epochs': 50, 'lr': 0.1, 'optimizer': <class 'torch.optim.adam.Adam'>}, Accuracy: 0.735
Params: {'dropout_rate': 0.7, 'epochs': 50, 'lr': 0.1, 'optimizer': <class 'torch.optim.sgd.SGD'>}, Accuracy: 0.79
Params: {'dropout_rate': 0.7, 'epochs': 100, 'lr': 0.001, 'optimizer': <class 'torch.optim.adam.Adam'>}, Accuracy: 0.74
Params: {'dropout_rate': 0.7, 'epochs': 100, 'lr': 0.001, 'optimizer': <class 'torch.optim.sgd.SGD'>}, Accuracy: 0.755
Params: {'dropout_rate': 0.7, 'epochs': 100, 'lr': 0.01, 'optimizer': <class 'torch.optim.adam.Adam'>}, Accuracy: 0.795
Params: {'dropout_rate': 0.7, 'epochs': 100, 'lr': 0.01, 'optimizer': <class 'torch.optim.sgd.SGD'>}, Accuracy: 0.78
Params: {'dropout_rate': 0.7, 'epochs': 100, 'lr': 0.1, 'optimizer': <class 'torch.optim.adam.Adam'>}, Accuracy: 0.77
Params: {'dropout_rate': 0.7, 'epochs': 100, 'lr': 0.1, 'optimizer': <class 'torch.optim.sgd.SGD'>}, Accuracy: 0.765
```

בחרנו את ה5 הכי טובים והמשכנו איתם לפרדיקציה משותפת

תוצאות ואבלואציה:

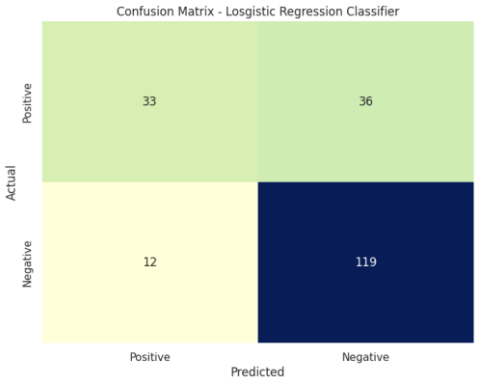
יש לציין ששלב זה מוצג כאן לאחר 10 איטרציות של CRISP-DM ועל כן אלו הם התוצאות אחריי יצירת פיצ'רים נוספים ועוד ניקוי דאטה כולל הוספת SMOTE ועוד דברים שפורטו בשלב העיבוד נתונים.

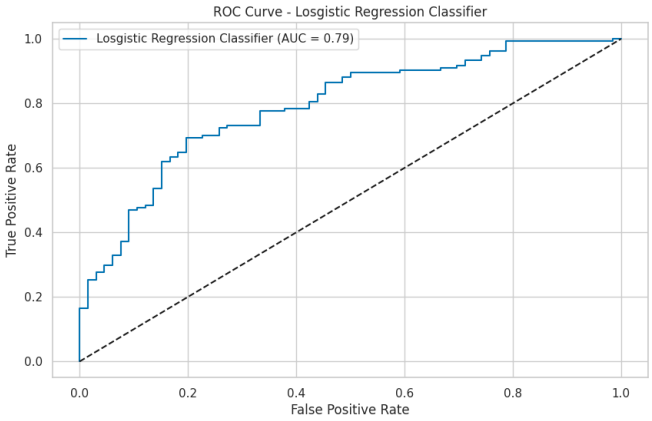
Logistic regression :

ביצועים ומטריקות:

```
                        Model  Accuracy  Precision   Recall   F-Score  \
Losgistic Regression Classifier     0.715   0.727811  0.91791  0.811881

 ROC AUC  Training Time
0.790932       0.031279
```

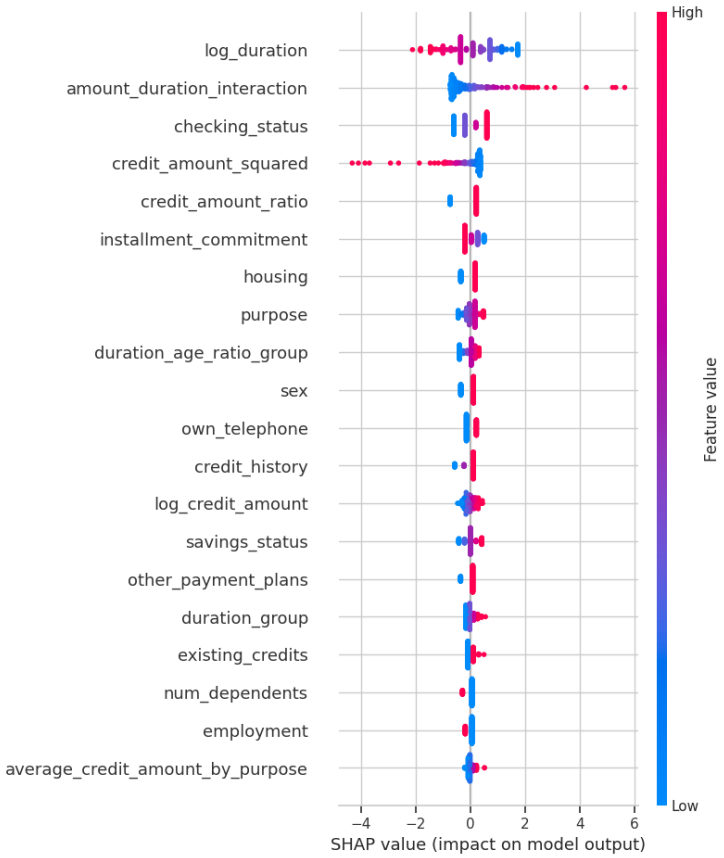




על פי הנתונים המוצגים בעמוד הקודם ניתן לראות כמה תובנות מעניינות על רגרסיה לוגיסטית.

ראשית כידוע לנו המודל הינו פשוט מידי ולא יכול לקחת בחשבון הרבה פיצ'רים, למרות שאנחנו עשינו כל מאמץ לעשות Feature engineering מתוחכם אנחנו עדיין לא הצלחנו להביא את המודל הפשוט הזה לרמה מספקת מכיוון שבצורה הכי פשוטה המודל הזה עושה Underfit משמעותי וזו הגבלה שקיימת בתוכו ואין כיצד לגשר על זה.

כאשר נתייחס למטריקות הראשונות נראה כי הAccuracy שלו הוא יחסית נמוך ולא הצליח להשתפר ואף החמיר ככל ששיפרנו את המודלים האחרים מה שמדגים את הפשטות של המודל ותומך בטענה שהוא לא יודע להתמודד עם כמות גדולה של פיצ'רים או פיצ'רים מתוחכמים כמו שיצרנו. נראה כי הRecall גבוה על חשבון הPrecision , זה מצב שמאוד נרצה להימנע ממנו כיוון שככל שנאשר הלוואות לא חכמות יותר ככה הסבירות של ההלוואה לא להיות מוחזרת עולה וככה הסיכוי לקריסת הבנק בסוף. מה שכן יאמר לזכותו של מודל זה הוא שהזמן אימון קטן מאוד וגם שהROC AUC שואף ל0.8 כך שההפרדה שלו בין הקלאסים אינה מאוד נוראית אך היא בהחלט ברמה לא מספקת. בConfusion matrix גם נוכל לזהות שיש לו נטייה חזקה לPositive כאשר אין סיבה לכך ואולי דבר זה נובע מכך שהTrain test התפלג בצורה לא אידאלית בכלל.

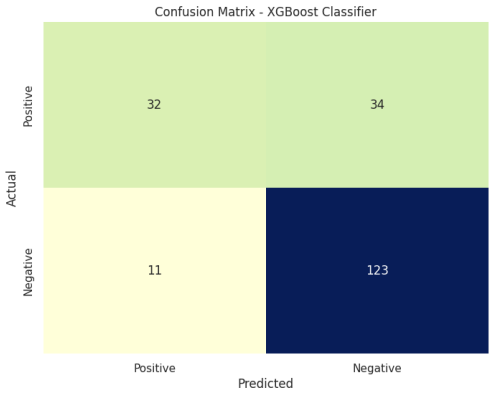


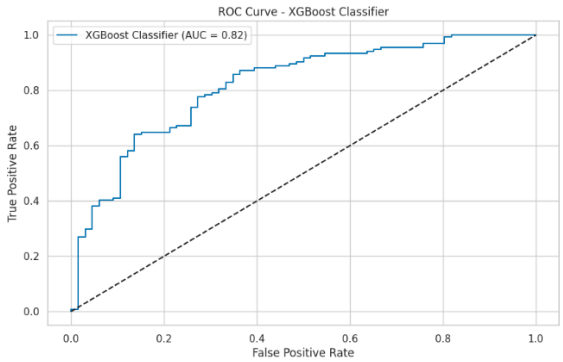
בנוסף לכך בShap ניתן לראות תוצאות מעניינות ולא צפויות, נראה שהמשתנים שמשפיעים מאוד על המודל זה ה(Log(duration שזה בעצם הלוג של הזמן החזרה, כאשר הפיצ'ר הזה נוצר על מנת לנרמל את הזמן החזרה בנוסף לסקיילר. כלומר ככל שהזמן החזרה יהיה יותר גבוה ככה עדיף להימנע מההלוואה כמה שיותר ולדוגמה גם שככל שהיחס בין הסכום הלוואה לזמן יותר גבוה אז ככה יש עדיפות לתת הלוואה. עוד דברים מעניים הם לדוגמה Checking status שמאוד מפוצל (כי הוא בעצם בבינים) אבל הוא מראה שככל שהוא יותר נמוך ככה אולי עדיף לא לתת את ההלוואה. גם נוכל לראות שככל שלמישהו יש יותר קרדיט אז אולי עדיף לא לתת לו את ההלוואה. נוסף לכך הרבה פיצ'רים שיצרנו נראה כי הם אינם רלוונטים.

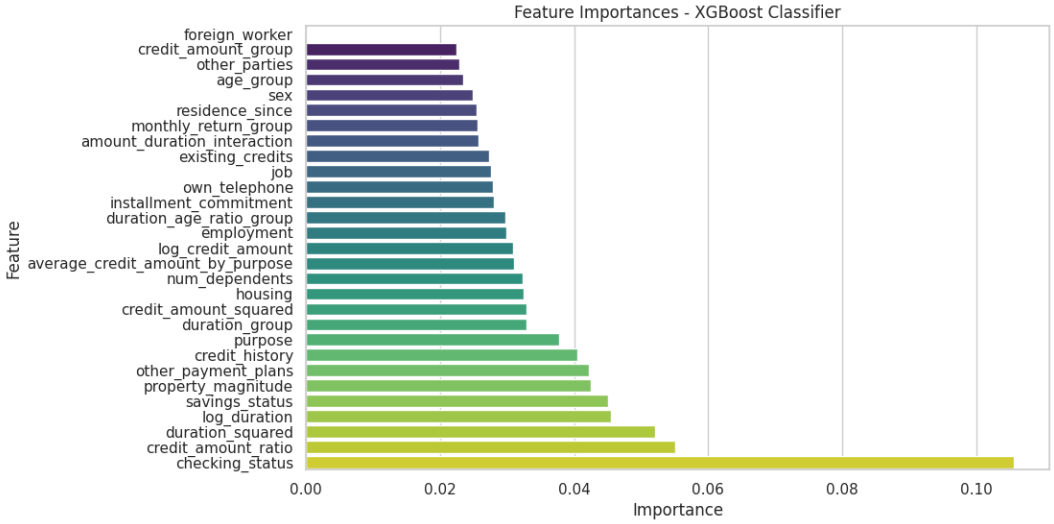
XGBoost:

ביצועים ומטריקות:

Model	Accuracy	Precision	Recall	F-Score	ROC AUC	Training Time
XGBoost Classifier	0.775	0.783439	0.91791	0.845361	0.818069	1.784477







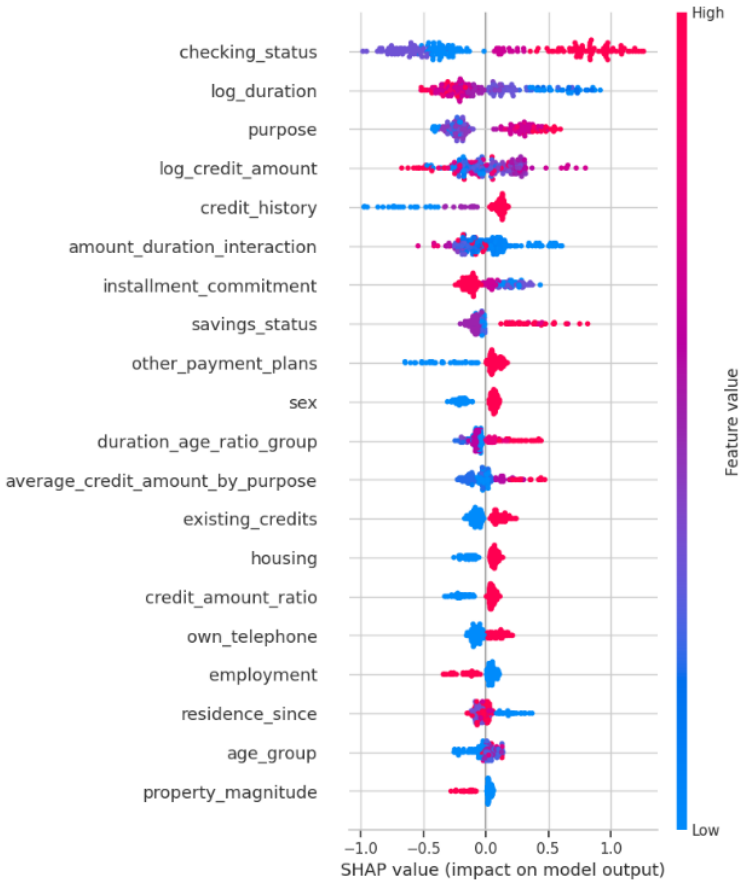
גם פה ניתן לראות ביצועים מעניינים מאוד.

אמנם גם כאן הAccuracy אינו טוב מספיק כיוון שאנחנו מקווים לפחות 80% על סמך הבייסליין אך כעת יש שיפור של 6% לעומת המודל הבסיסי של הרגרסיה הלוגיסטית. זה שיפור משמעותי שכבר מראה על חולשת המודל הקודם. נוסף על כך המודל הזה מדגים לנו כמעט את אותו הRecall אך הPrecision השתפר משמעותית מה שמצביע על כך שהמודל כעת יותר מדויק בתפיסת המקרים שאכן יש לאשר הלוואה.

יותר מכך הF-Score השתפר אשר מצביע על יחס יותר טוב בין הPrecision לRecall משמע אנחנו מתקדמים בכיוון שאנחנו רוצים ומצפים אך שילמנו על כך בזמן אימון אך לא משמעותי מידי. נוסף על כך כעת ה ROC AUC השתפר משמעותית וכעת ניתן לראות 82% כלומר הפרדה משמעותית יותר טובה בין הקלאסים שזה בדיוק מה שאנחנו רוצים במטלה כזאת.

כעת כיוון שגם השתמשנו בסוג של מודל מורכב מסוג Boosting גם קיבלנו את ההשפעה של כמה מהפיצ'רים על ההחלטה שלנו, דבר שהינו מאוד קריטי למקבל ההחלטות בבנק. ניתן לראות כי בהשוואה למודל הקודם הChecking status כעת הוא המשפיע ביותר ולאחריו הCredit amount ratio שזה בדיוק מה שאנחנו מצפים לראות כאשר אנחנו נותנים הלוואה, מאוד חשוב לנו להבין האם הוא הלקוח החזיר את ההלוואות הקודמות

והאם היו לו בעיות כולשהן בתשלום. עקב כך אני יכול להסיק שהמודל הזה מבין את הפיצ'רים בצורה הרבה יותר מורכבת ומעניינת וכרגע הוא הפך להיות הבייסליין שלנו באיטרציה הנוכחית.



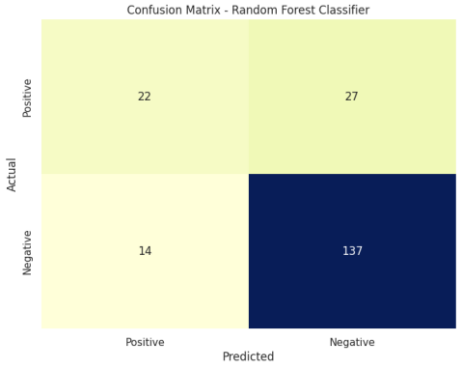
נוסף על כך הShap כעת עוזר לנו להבין יותר את המורכבות של המודל בקבלת ההחלטה והוא נראה לנו הרבה יותר רלוונטי מאשר הקודם כאשר באמת ניתן לראות את החשיבות של Checking status וההפרדה המאוד קשוחה בינהם וגם את הLog duration שהוא גם משפיע מאוד עיקרי. כעת פיצ'רים מורכבים להבנה שקיימים בדאטה המודל מצליח לתפוס בצורה לא רעה בכלל כגון המטרה של ההלוואה (Purpose) שהוא בעצם פיצ'ר קטגוריילי אך כידוע לנו הבנק הרבה פעמים ישאל מה המטרה בעצם של ההלוואה ואם היא לא מטרה ראויה אז יכול לדחות אותנו. עקב כל מה שפורט כאן המודל הזה מציג תוצאות מאוד מעניינות לפרוייקט שלנו והבנו גם כיצד ניתן להתקדם.

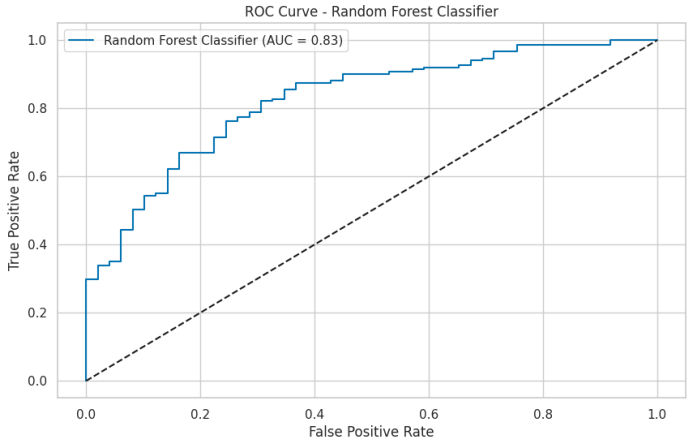
:Random forest

מודל זה מאוד מעניין אותנו כיוון שזהו מודל Ensemble במיוחד עבור מטלה זו. כל עץ על סמך הפיצ'רים שהוא החליט ירכיבו לנו את ההחלטה הסופית ובעצם כך נקבל מודל מאוד מאוזן ושאמור לבצע את המטלה הזאת בצורה נפלאה. זאתי הסיבה שבחרנו אותו בתור מודל ראשוני גם.

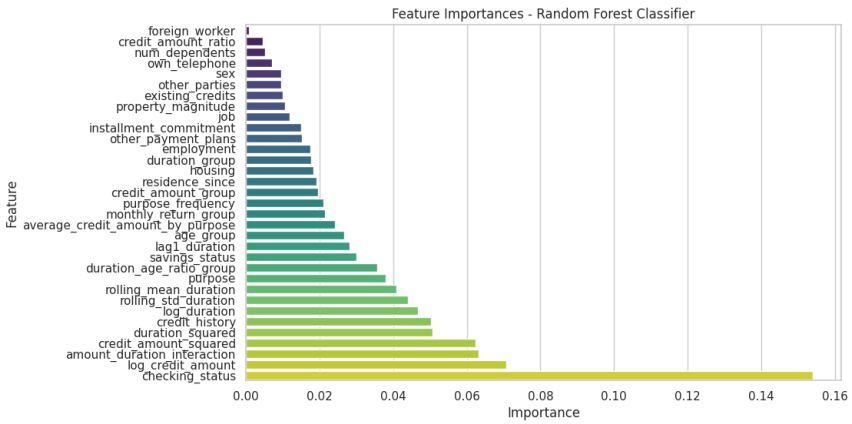
תוצאות ומטריקות:

Model	Accuracy	Precision	Recall	F-Score	ROC AUC	Training Time
Random Forest Classifier	0.795	0.835366	0.907285	0.869841	0.825247	1.290269

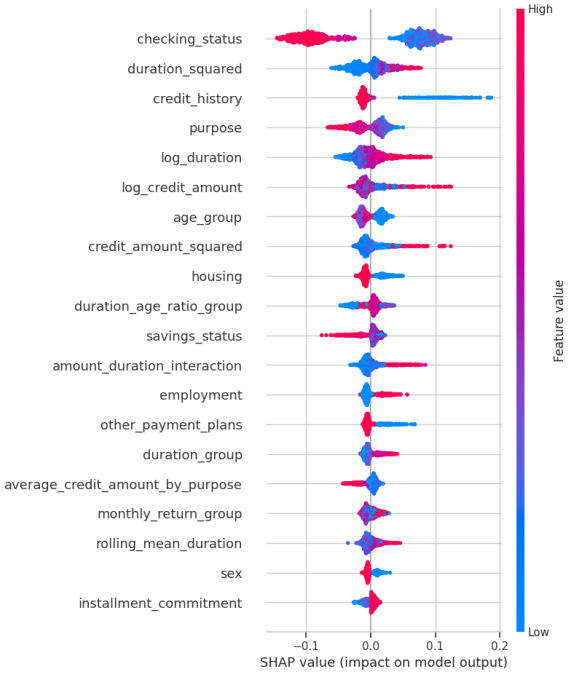




התוצאות גם כאן היו מאוד מעניינות. זכינו לראות שיפור משמעותי (שבא על חשבון הזמן אימון) בין המודל של הXGBoost לבין הRandom forest כלומר ניקח את זה כרגע כבייסליין שלנו להמשך העבודה. הAccuracy כאן השתפר משמעותית בכ-2% ואומנם זה נראה לא משמעותי אך זה מכוון אותנו בכיוון של להבין שEnsamble עלול מאוד להתאים למטרה בהינתן הפיצ'רים שקיימים אצלנו. יותר מכך הPrecision עלה שזה מאוד משמעותי בשבילנו מהסיבות שהסברנו ואכן זה בא על חשבון ה Recall אך כמו שכבר הגדרנו בהתחלה זה משהו שלא ניבהל ממנו ואפילו זה טוב לנו כיוון שהמטלה היא לא מטלה ברפואה שמצדיקה Recall גבוה כלומר לתפוס את כל החיוביים אלא מעדיפה לתפוס רק את החיובים ולשגות כמה שפחות. ה Fscore עדיין מצביע על איזון טוב. יותר מכך ניתן לראות שהRoc AUC השתפר משמעותית וזה בדיוק מה שנרצה לראות ! הוא מצליח להפריד בין הקלאסים השונים בצורה הרבה יותר טובה מהמודל הטוב הקודם שלנו ועל כן זה בדיוק מסוג המודלים שאנחנו נרצה להתמקד בהם ובהמשך גם נבנה עוד מודל על סמך ההבנה הזאת.



נוסף על כך גם נקבל את חשיבות הפיצ'רים עבור מודל זה ושוב נזהה שהChecking status מאוד חשוב בניגוד למודלים כמו LogReg , כלומר המודים האלה הרבה יותר טובים בלקחת חשבון פיצ'רים שאולי ההבנה שלהם לתרומת המודל מורכבת מידי למודלים פשוטים. בדיוק כמו בXGBoost ראינו יותר חשיבות לפיצ'רים כאלה.



גם בעזרת הShap כעת נוכל לזהות את החשיבות של כל אחד מהפיצ'רים ונראה שוב בדיוק כמו בXGBoost התייחסות מאוד חשובה לChecking status (יש לציין שהצבעים הפוכים במקרה זה כיוון שהמודלים נוצרו באיטרציות שונות ועל כן פעם אחת הLabel encoder נתן להם תיוג אחר מהתוצאה הראשונה ועל כן אנו רואים צבעים שונים, אסביר על זה בהרחבה בסיכום ואיך נתמודד עם זה כדי שהXAI יהיה משמעותי עבור מקבל ההחלטות). אך למרות כל זה, המודל עדיין נותן לנו הבנה יותר מעמיקה לפיצ'רים ונראה כמה הChecking status לדוגמה קריטי כי הוא מופרד בצורה מוחלטת כמעט. גם בCredit history ניתן לראות כמה הוא משפיע כאשר הוא מתוייג יותר נמוך. כמעט בכל אחד מהפיצ'רים ניתן לראות הפרדה מאוד חזרה לחיוב או לשלילה ואומנם הם לא משפיעים עם המודל במידה של Checking status הם עדיין משפיעים וכאשר כולם נלקחים בחשבון זה בדיוק מה שנותן את ההחלטה הטובה האם לתת הלוואה או לא. בעצם קיבלנו תובנות מאוד טובות מהXAI וככה ידענו להתקדם באיטרציות בCrisp-DM . בתוצאות הנוכחיות אני דיי מבסוט בשלב זה ועל כן אחריי מימוש מודל זה אני אמשיך למודלים יותר מורכבים.

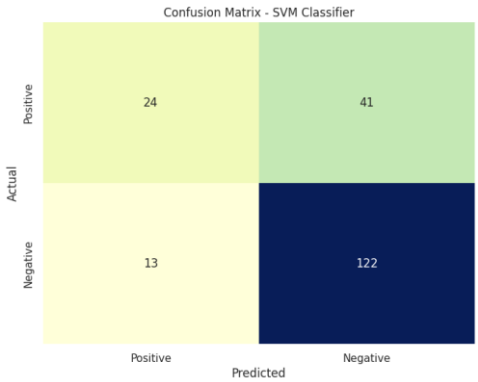
: SVM (SVC)

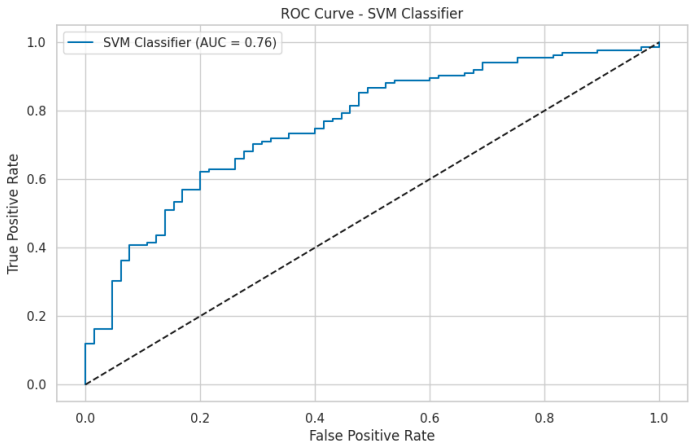
בחרנו במודל זה גם כיוון שידוע לנו מקורסים קודמים שזה מודל מאוד טוב להפרדה במרחב הפיצ'רים בין קלאסים שונים ועושה את זה בצורה מאוד מתוחכמת ואומנם הוא לא טוב כמו בוסטינג או אנסמבל למינהם עדיין נרצה לבחון אותו כי אולי הבעיה יותר פשוט ממה שנדמה.

תוצאות ומטריקות:

```
          Model  Accuracy  Precision    Recall   F-Score   ROC AUC  \
SVM Classifier      0.73   0.748466  0.903704  0.818792  0.761823

Training Time
      0.30784
```





בצורה דיי מפתיע הSVM לא הציג ביצועים טובים בכלל, אמנם הRecall גבוה כלומר הוא מכל החיוביים שהוא תפס הוא הצליח באמת לתפוס את רוב החיוביים אך זה בא לנו מאוד על חשבון הPrecision שמציג תוצאות מאוד לא טובות ועל חשבון הAccuracy ולאחר שבחרנו להתקדם עם Random forest כמודל המועדף כעת, נראה שSVM הוא ברמה של הLogistic regression . אני לא בטוח כיצד להסביר את התופעה המאוד מוזרה הזאת אך כנראה זה נובע מכמות מאוד נמוכה של פיצ'רים נומרים והרבה פיצ'רים כנראה לא מצליחים להסביר את המורכבות של המטלה. המודל גם מציג Roc AUC מאוד נמוך ועל כן אין סיבה להמשיך לבחון את המודל.

(יש לציין שכרגע במחברת אחריי שינויים מאוד מינורים הSVC מציג תוצאות יותר טובות של Accuracy 0.79 וPrecision של 0.84 וROC AUC של 0.81. אך עדיין בהשוואה לRandom forest התוצאות לא טובות ועל כן לא אסביר מחדש על כך.

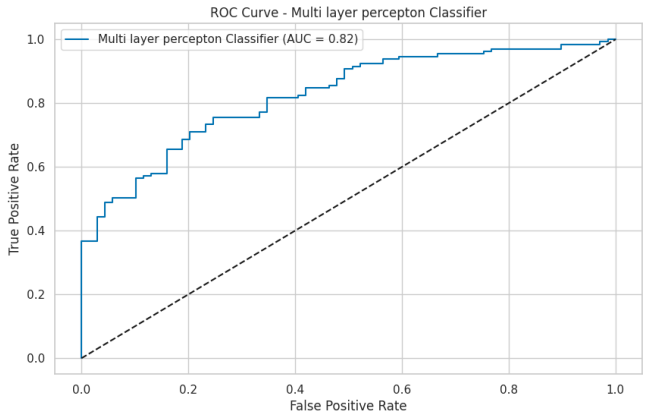
:Multi layer perceptron

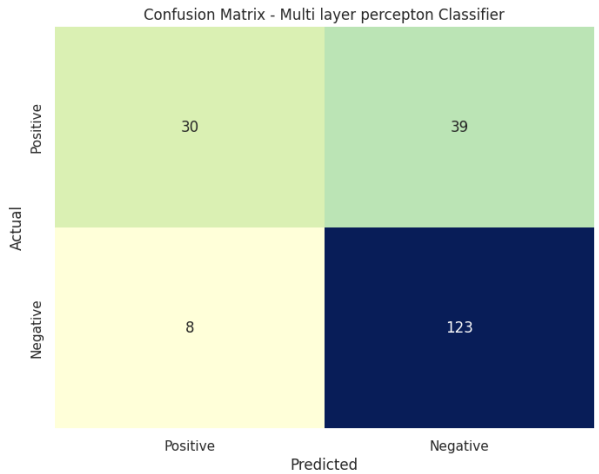
בחרנו במודל זה כמעין POC עבור הכוח של רשתות ניורנים (לא עמוקות בשלב זה) ועל כן בהיותו POC הוא מאוד מנוון ופשוט אך הוא ידגים לנו בתקווה האם כדאי להמשיך עם מודל מסוג זה. ניקח בחשבון שהבעיה הכי גדולה במודלים מסוג זה היא החוסר בXAI ומאוד קשה להסביר למקבל ההחלטות את הבחירה של המודל ועל כן בבעיה מסוג זה אנחנו לא נרצה לפתור את זה כך. אך אם הביצועים של המודל הזה הם משמעותית יותר טובים נעדיף להתקדם איתו.

תוצאות ומטריקות:

```
                                Model  Accuracy  Precision    Recall  F-Score  \
Multi layer percepton Classifier     0.765   0.759259  0.938931  0.83959

 ROC AUC  Training Time
0.823653       4.734415
```





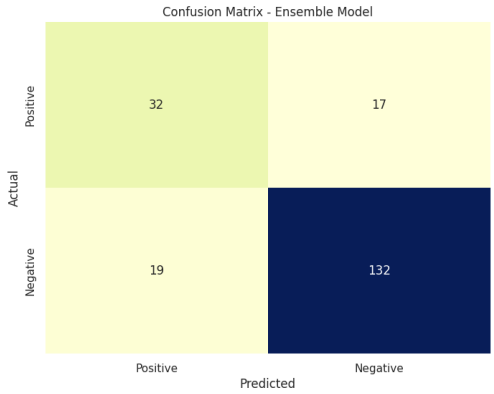
ואכן נראה כי המודל זה קיבל תוצאות מאוד מעניינות. דבר ראשון נראה כי הAccuracy שלו הוא משמעותית יותר נמוך מהמודלים שכעת אנחנו מסתמכים עליהם אך ניתן להבין בכמה דברים מעניינים: דבר ראשון ה Roc AUC שלו גבוה יחסית ועל כן יש לו יכול הבחנה בין הקלאסים השונים יחסית טובה שזה מעניין ומצביע על כך שרצוי לבחון את המודל יותר לעומק. יותר מכך הRecall שלו מאוד גבוה כלומר הוא ניסה לתפוס יותר את כל הרשמות שצריך לתת להם הלוואה, מה שנובע כנראה מUnderfit ונטייה ל"לתת הלוואה" ואכן בConfusion matrix נוכל לראות כי זה אכן מה שקורה ויש העדפה משמעותית לPositive . חשוב לציין שהPrecision עדיין מאוד נמוך ועל כן רצוי לבצע מודל מסוג זה אך יותר מורכב

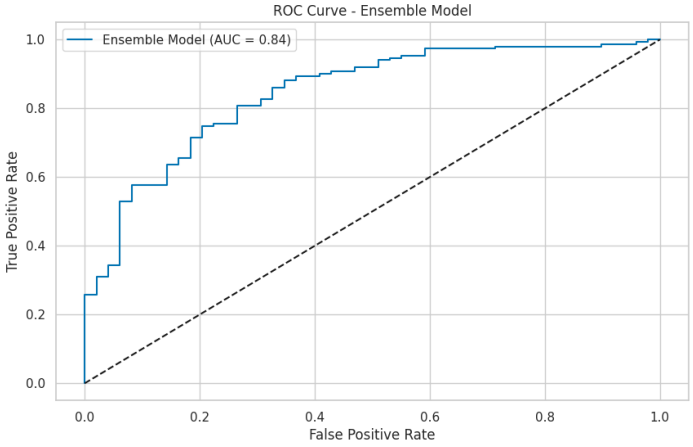
: FFNN ensemble

לפי התובנות לאורך כל העבודה אנחנו רואים כי שיטות Ensemble מציגות תוצאות טובות וגם הרשת ניורונים המנוונת הציגה תוצאה מעניינת שנרצה לבחון על כן מה שמתבקש זה לעשות שיטת Ensabmle עבור רשתות ניורנים ובכך לנצל את הכוח של שתיי השיטות שעבדו בצורה הטובה ביותר.

תוצאות ומטריקות:

	Model	Accuracy	Precision	Recall	F-Score	ROC AUC
0	Ensemble Model	0.82	0.885906	0.874172	0.88	0.844438





בעצם קרה כאן משהו מאוד מעניין. נבחן את התוצאות על מנת לראות זאת.

כעת הAccuracy שלנו הוא על 82% אחוז שזה יותר טוב מהRandom forest שהציג לנו 80% לכל היותר ועל כן בפן הזה המודל כבר יותר טוב. אך יותר חשוב מזה ניתן לראות את הFscore שעומד על 0.88 ואת הPrecision שגם הוא עומד על 0.886 , כלומר כעת קיבלנו את המודל שאליו שאפנו. המודל הזה מציג ביצועים טובים מאוד מבחינת הPrecision כלומר הוא הTrue positive שלו מתוך כלל החיזויים לPositive הם 88% , משמע המודל עושה החלטות הרבה יותר חכמות האם לתת הלוואה או לא. זה מדגים לנו שהמודל הוא צודק בהאם לתת הלוואה ב88% הפעמים, אנו בעצם הורדנו את הסיכון של לתת הלוואה כשלא צריך. יותר מכך יש Recall של 87% שזה יותר נמוך מהמודלים הקודמים אך בעצם זה אומר לנו שהוא חזה 87% את המקרים של לתת הלוואה מתוך כל המקרים שהיה צריך לתת הלוואה, וכמו שציינו מקודם אין לנו שום בעיה עם מודל שלא תופס את הכל אלא תופס באמת שאת מה שצריך (בניגוד למודלים בתחום הרפואה). יותר מכך הRoc AUC הוא הכי גבוה שהתקבל עד כה כלומר המודל הזה יודע להפריד בין הקלאסים בצורה הרבה יותר טובה. זה בול המודל שחיפשנו לאורך כל העבודה. לצערנו בשלב זה אין XAI מספיק טוב למודלים מסוג זה ועל כן נסתפק במודל הזה בלי לדעת באמת את תהליך קבלת ההחלטות שלו.

הצעות להמשך:

כמובן שהפרוייקט הזה אינו מושלם ולא יכול להתמודד מול מערכות State of the art אבל ישנם כמה שיפורים שלעניות דעתי יאפשרו עוד התקדמות מבחינת המטריקות אשר חשובות לנו.

1. יהיה מאוד מעניין לבחון מודל הרבה יותר מורכב מהסוג הבא: ליצור מעין Autoencoder אשר ינסה למפות כמות פיצ'רים למימד הרבה יותר מצומצם. על מנת לעשות זה נצטרך לאמן Autoencoder ולנסות כל פעם לבצע שחזור לכך ואחריי שהמודל יהיה מאומן נעשה לו כתימה של השכבות האחרונות ובכך בעצם נחלץ את הEmbeddings של המודל נחבר לבסוף רשת נוירונים פשוטה כמו שיצרנו בפרוייקט זה וננסה לבצע את המטלה מחדש על סמך הAutoencoder. לדעתי זה יציג שיפור משמעותי כיוון שכעת המודל יהיה יותר רובסט לשינויים וייקח בחשבון את כל המרחב Embedding . בעיה שקיימתי בדאטה סט אז היא הנרמול וכמות המידע מה שעלול מאוד להפריע למודל מסוג זה ועל כן בפרוייקט הנ"ל לא פיסיבילי אך כיוון המשך מאוד מעניין
2. בהמשך להצעה הקודמת אפשר להוסיף בסוף אפילו Siamese network ולעשות מעין Pattern recognition כך שבכל פעם נעשה השוואה בין קלט שראוי להלוואה וקלט שלא. מודל מסוג זה יצטרך המון מידע מנורמל של לקוחות והאם ניתנה להם הלוואה אך לדעתי אם ניתן לו כל פעם קלטים שונים של רשומות שצריך לתת להם הלוואה ורשומות שלא נוכל לקבל תוצאות מאוד טובות. רצוי אפילו להשתמש בTriplet loss עם רשומה Anchor שהיא אידאלית רשומה Positive שנדגמת מהדאטה ורשומה Negative
3. לפי ההצעות הקודמות אנו מבינים שיש בעיה בכמות דאטה ובסידור שלו. בהתייעצות עם אנשים מתחום הפיננסים אפשר לעשות Generated data כלומר דאטה שהוא לא מהמקור אך יוצרים אותו כדי לנפח את כמות הנתונים והוא נאמן למקור (טכניקה נפוצה ב Computer vision אך ששם זה נושא יחסית קל למימוש)
4. עוד הצעה שיכולה להועיל מאוד ולא לעלות הרבה היא לזנוח את הדאטה הזה ולגשת לבנק יותר גדול ולבקש ממנו את הסט נתונים של ההלוואות (בהנחה שהוא יחסית מנורמל בצורה יותר טובה) ובכך בעצם גם להקל על הפייפליין שלנו שמנרמל את הדאטה ולהוריד מהזמן עבודה של הקוד.
5. רצוי לחקור עוד דברים שעלולים להשפיע, כגון פרופיל הבן אדם שלא נמצא כאן אך אני כמעט בטוח שהבנק מאפשר לעצמו לקחת עוד פרטים על הבן אדם שנחשבים לתחום אפור. רצוי לקחת נתונים כמו צבע עור ורקע משפחתי וכו. יכול להיות שיש Bias מסויים שמשפיע על ההחלטה וצריך לנטרל אותו אך תחילה צריך להבין אם הוא קיים. כמובן במודלים שיצרנו כאן, אם נכנס Bias מסויים בגלל הקלאס שהפקיד בנק תייק אז אין לנו איך לדעת את זה ויכול להיות שזה מאוד מטה את המודל והוא לומד דברים שהוא לא צריך ללמוד. ההחלטה להלוואה אמורה להיות נטרלית ועל הסמך הבן אדם והיכולות החזרה בלבד ולא עוד פיצ'רים שאינם רלוונטים ובני אדם שוגים בהם הרבה.
6. צריך לעשות ניקוי פיצ'רים לדאטה הקיים כאן, לדוגמה יש לנו את משתנה הSex שלעניות דעתי אינו רלוונטי בשום צורה לקבלת החלטה האם לתת הלוואה אך בכל אחד מהמודלים קיבלנו שהפיצ'ר הזה הוא בעל חשיבות מסויימת ונמצא בדירוג ה9 לפי Shap. לדעתי זה מאוד חריג וצריך לבחון השפעות כאלה על המודל.
7. הרבה מהפיצ'רים לא מוסברים מספיק כי הם קטגוריילים ופשוט הLabel encoder נתן להם מספר. צריך לחשוב על דרך טוב לייצג אותם ולא בסתם מספר כיוון שניתן לראות בפיצ'רים כמו Sex שזה קטגוריילי בינארי שבSHAP הוא נתן לנו שככל שהוא יותר גבוה ככה זה יותר טוב למודל. אך מה זה בעצם גבוה? אני בחנתי את זה וראיתי שMale זה יותר גבוה אך למקבל ההחלטות אין שום הבנה בסט נתונים עצמו והפיצ'ר הזה בעצם לא יסביר לו כלום. צריך ליצור מעין Label encoder משלנו לפי ההבנה הזאת אך צריך גם לעשות זאת בצורה חכמה.
8. המון נתונים שלדעתי רלוונטים חסרים כגון מצב עובר ושב, הכנסה חודשית (ולא סתם Stable , פיצ'ר נומרי יתרום לנו משמעותית יותר), האם הלקוח נמצא במינוס (הפיצ'ר הזה אומנם יכול להיכנס תחת הCredit אבל לדעתי צריך לפלג את זה כדי לתרום להבנה), מה הדירוג אשראי של הלקוח?
כל אלה הם מאוד חשובים והמודל לא יגיע לביצועים טובים בחיים בלעדיהם.

סיכום:

כאשר התחלנו בפרויקט זה, המטרה העיקרית שלנו הייתה לפתח מודל סיווג שיוכל לקבוע במדויק האם יש לאשר בקשת הלוואה של לקוח על סמך סיכויי ההחזר שלו. משימה זו רלוונטית מאוד בתעשייה הפיננסית של ימינו, שבה תחזיות מדויקות יכולות להפחית משמעותית סיכונים ולשפר את תהליכי קבלת ההחלטות, במיוחד בהתחשב במקרים היסטוריים כמו המשבר הפיננסי של 2008, שבהם החלטות לקויות בנושא הלוואות הובילו להשלכות קטסטרופליות.

במהלך הפרויקט, עסקנו במערך נתונים שאמנם היה פשוט לכאורה, אך דרש עיבוד מוקדם והנדסת פיצ'רים מהותית. מערך הנתונים כלל 1,000 רשומות, כל אחת עם 20 תכונות, אך הנתונים היו רחוקים מלהיות נקיים. האופי הבלתי מאוזן של מערך הנתונים - 700 מקרים מתויגים כ"טובים" ו-300 כ"רעים" - היווה אתגר משמעותי, שהצריך שימוש בטכניקות מתקדמות כמו שיטות אנסמבל ורשתות נוירונים כדי להבטיח ניבוי טוב.

הגישה שלנו כללה תהליך איטרטיבי שבו נרמלנו את הנתונים, הנדסנו פיצ'רים חדשות והתנסנו במודלים שונים, כולל רגרסיה לוגיסטית, XGBoost, SVM ו- Neural Networks. בשלב מוקדם של התהליך, תצורות מודל ברירת המחדל הניבו תוצאות מבטיחות, שכבר עלו על דיוק כמה מהמדדים ב-Kaggle. זה הדגיש את החשיבות של עיבוד נתונים נכון, ולקחנו את התובנה הזו קדימה על ידי כוונון של המודלים שלנו.

חקרנו את כוחן של שיטות אנסמבל, במיוחד ברשתות נוירונים, וגילינו ששילוב של מספר מודלים שיפר את התוצאות שלנו. דגמי האנסמבל הראו את הביצועים הטובים ביותר, עם דיוק שהגיע ל-82% ומאזן Recall וPrecision שהפחית משמעותית את הסיכון לאישור הלוואות רעות.

השימוש בערכי SHAP וניתוח חשיבות תכונות סיפקו לנו הבנה מעמיקה יותר של התכונות שהשפיעו ביותר על תהליך קבלת ההחלטות. באופן מעניין, תכונות כמו שינוי הDuration שהפקנו ממנו עוד הרבה פיצ'רים, הפיצ'ר של הCredit to amount ratio ועוד הופיעו כמנבאים קריטיים, וחיזקו את החשיבות של הנדסת פיצ'רים זהירה. רוב הפיצ'רים המשמעותיים עבור המודלים שלנו היו הפיצ'רים שאנחנו בעצמו יצרנו !!!

לסיכום, פרויקט זה לא רק אפשר לנו ליישם טכניקות למידת מכונה מתקדמות אלא גם הדגיש את הערך של פיתוח מודלים איטרטיביים מונעי דאטה ובדיוק הראה לנו את החשיבות של מה שלנו עוד בשיעורים הראשונים כגון CRISP-DM. התובנות שהושגו מניתוח ה SHAP וחשיבות הפיצ'רים עזרו לחדד את המודלים שלנו ולשפר את יכולת ההבנה שלהם. בהמשך, היישום של מודלים מורכבים יותר והכללת תכונות נוספות עשויות לשפר עוד יותר את הדיוק והאמינות של תחזיות אישור ההלוואה.