

תוכן עניינים

מבוא.....	1
3..... עיקרי השינויים חלק א	1.1
3..... מערכת ניירונים – Neural Networks.....	2
3..... הרצת הרשת בעזרת ברירת המחדל.....	2.1
4..... כוןנו פרמטרים.....	2.2
4..... קונפיגורציות.....	2.3
6..... Decision Trees – עץ החלטה	3
6..... בנייה עץ החלטה באמצעות סט אימון ואמות	3.1
6..... הרצת פקודת ()printcp.....	3.2
8..... קטימת העץ לפני ערכיו cp נוספים	3.3
8..... השוואה בין חמישת המודלים	3.4
9..... K - Means	4
9..... הוצאת משתנה המטריה מתוך ה <input/>	4.1
9..... הרצת מודל עם ערכי brt מחדל	4.2
9..... טיב ההתאמה בין האשכולות למחלקות	4.3
10..... ערך A מיטבי	4.4
11..... השוואה בין המודלים	5
13..... המודל הנבחר.....	6
13..... צגת המודל הנבחר	6.1
13..... ניתוח תוצאות בעזרת מטריצת מבוכה	6.2
13..... הגשת חיזויים סופיים	7
14..... נספחים	8
14..... הכנות הנתונים	8.1
14..... נספח 1 – עיקרי חלק א'	8.1.1
14..... נספח 2 – חלוקה לסט אימון ובחינה	8.1.2
14..... נספח 3 – חלוקה לסט אימון ואמות	8.1.3
15..... נספח 4 – הרמת משתנים קטגוריאליים למשתני דמה	8.1.4
16..... רשת ניירונים	8.2
16..... נספח 6 – הרצת הרשת עם נתונים בירית מחדל	8.2.1
16..... נספח 7 – ערך אופטימאלי קונפיגורציה decay	8.2.2
17..... נספח 8 – מספר משקלות נלמדות בקונפיגורציות השונות	8.2.3
17..... נספח 9 – ערך אופטימאלי קונפיגורציה מספר איטרציות	8.2.4
18..... נספח 10 – מספר איטרציות אופטימאלי לערך decay נתון 0.85	8.2.5
18..... נספח 11 – ערך decay אופטימאלי למספר איטרציות נתון 40	8.2.6
19..... נספח 12 – עץ החלטה לפניה קטימה	8.2.7
19..... נספח 13 – עץ החלטה לאחר קטימה לפני ערך cp אופטימאלי	8.2.8
19..... נספח 14 – פלט העץ לאחר קטימה בעזרת שימוש בplot.rpart	8.2.9
20..... נספח 15 – קשר בין מאפיין outcome למשתנה המטריה ו שנמצא בחלק א'	8.2.10
20..... נספח 16 –ניתוח מאפיינים שונים בתור מסווגים	8.2.11

1 מבוא

בחלק זה, נשתמש בתנאים אשר סקרנו והכנו בחלק הקודם לצורך אימון ובחינה של מערכות לומדות והשוואה ביניהן. מטרת הפרויקט הינה ביצוע תחזית לגבי הצליפות&א הצליפות של לקוחות לתוכנית פיקדון.

1.1 עיקרי השינויים חלק א

חלק מעיבוד הנתונים בחלק הקודם, החלנו להוריד שלושה משתנים – gender, default, contact. בנוסף, את חודשי השנה במשתנה month בחרנו לאחד לפי רבעונים. את משתנה days בחרנו להציג באופן קטגוריאלי בחלוקת ארבע קבוצות – (1) לא נוצר קשר, (2) נוצר קשר בחצי השנה האחרון, (3) נוצר קשר בין חצי שנה לשנה אחרת, ו- (4) קשר אחרון נוצר לפני מעלה משנה. מבחינת נתונים חסרים, ביצענו הלמה בעזרת אלגוריתם mice גם נתונים האימון וגם נתונים הבדיקה ([תוספת 1](#)). ביצענו חלוקה ל-2 סטים – סט אימון וסט בבחינה. את החלוקת ביצענו בצורה רנדומלית כך ש-80% מהערכים הם בסט האימון ו-20% בסט בבחינה. ([תוספת 2](#)). בנוסף, השתמשנו בשיטת k-fold לחולקת סט האימון ([תוספת 3](#)).

2 רשת נירוניים – Neural Networks

לפני תחילת האימון בעזרת רשת נירוניים, יש צורך להתמודד עם המשתנים הקטגוריאליים. את כל המשתנים אלו, הגדרנו בעזרת משתני דמה (לפי קידוד 1,1-) ([תוספת 4](#)) והפעלו את פונקציה () as.factor ב כדי להגיד אותם כמשתנים קטגוריאליים ולא כערך נומי. בנוסף, הפעילו את פונקציית () scale על המשתנים הרציפים, ב כדי לנורמל אותם לאותה סקלת ערכים וסדר גודל.

2.1 הרצת הרשת בעזרת ברירת המחדל

ערci בברירת המחדל הם 22 נירוניים בשכבה הcnisah (מספר המאפיינים), שכבה חבויה אחת בעלת ניירון אחד ושכבה יצאה בעלת 2 נירוניים ([תוספת 5](#)). למעשה שכבת הcnisah והיציאה הין קבועות, ואילו מספר השכבות החבויות, מספר הנירוניים בכל שכבה צאת הוא היפר-פרמטר שיש לקבוע את ערci (בחבילת netach קיימת שכבה חבויה אחת). ישן 27 Möglichkeiten שהרשת צריכה לקבוע בהינתן ערci קונפיגורציה זו:

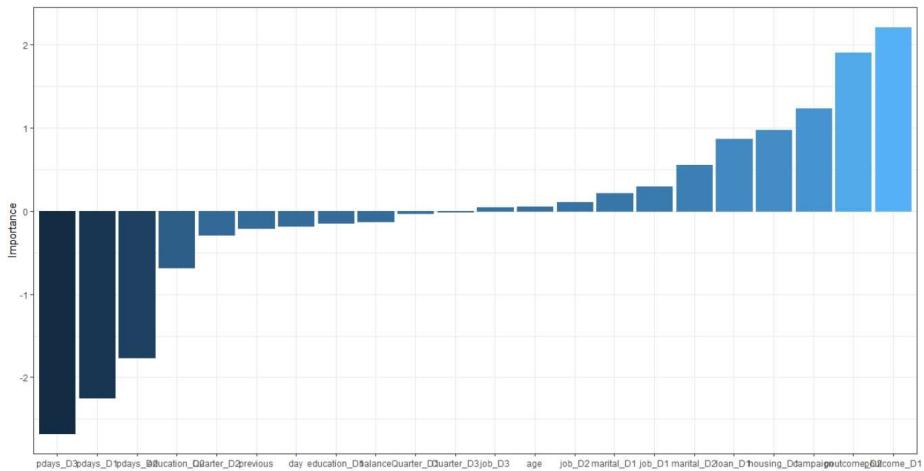
$$22 \text{ biases} = 27 \text{ weights} + \text{bias}_1^2 + \text{bias}_2^2 + \dots + \text{bias}_{22}^2$$

חישבנו את אחוז הדיק בשלב האימון והבחינה ב כדי לראות האם קיימtz מצב של over/under fitting.

test	train
0.76505	0.77358

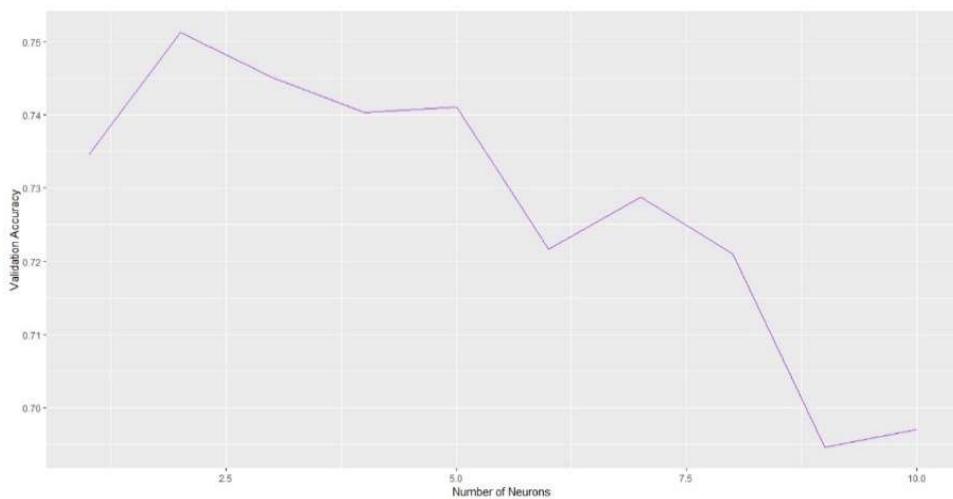
ניתן לראות כי קיבלנו הפרשים קטנים מאוד בין אחוז הדיק. מדד הדיק עבר סט האימון גובה 8.53×10^{-3} אחוזים מאשר הדיק עבור סט הבדיקה. לעומת, אנו לא נמצאים במצב של over fitting.

בנוסף, יצרנו תרשימים `olden` אשר מתאר בצורה ויזואלית את השפעה של כל מאפיין על אחוז הדיק. ניתן לראות כי למאפיין `outcome` (תוצאת הקמפיין הקודם) ישנה השפעה חיובית חזקה מאוד על אחוז הדיק. בנוסף, למאפיין `days` (כמה זמן עבר מהתקשרות الأخيرة עם הלוקוח) ישנה השפעה שלילית חזקה על אחוז הדיק. שתי מסקנות אלו מתיישבות עם מה שראינו בחלק א' – ראיינו כי למאפיינים אלו יש מתאם חזק (חיובי/שלילי) בהתאם (עם המשתנה התלוי). השפעה שלילית חזקה (אחוז השפעה סביר אפס) ניתן לראות עבור המאפיינים – עבודה, רביעון וגיל.



2.2 כוונון פרמטרים

ביצענו תהליך של כוונון פרמטרים – מספר הנוירונים בשכבה החבוייה, בניסיון להעלות את אחוז הדיק של הרשת. נרצה למצוא את מספר הנוירונים בשכבה החבוייה אשר ימקסמו את אחוז הדיק של סט האימוט. נציג גרפף של אחוז הדיק כפונקציה של מספר הנוירונים בשכבה החבוייה. השתמשנו בשיטת k -fold על מנת לבדוק את ערכי הפרמטרים שיביאו למקסום דיק של סט האימוט. הערכים שבדקנו נעים בין 1-101 בקפיצות של 5 וראינו כי המקסימום מתקיים בערכים יותר נמוכים ולכן התמקדנו בערכי 1-10 בקפיצות של 1.



ניתן לראות כי בהתחילה ישנה עלייה באחוז הדיק, כי הוספה הנוירונים בשכבה החבוייה הופכת את המודל למורכב יותר, וכך המודל מבצע חיזוי בצורה טובה יותר גם על סט האימוט. אך ברגע שעוברים את מספר הנוירונים האופטימאלי, מקבל כי האימון מותאם יותר לנוטוי האימון ופחות לאחוז הדיק של האימוט כך שהם יילכו וירדו, כי המודל לא מצליח לבצע חיזוי בצורה טובה לנוטוי האימון עליהם, הוא התאים את עצמו יותר מדי לנוטוי האימון ("קפיצות", שועלות להיגרם כתוצאה מאילוצים שונים של פרמטרים של הרשת, למשל מספר איטרציות מקסימאלי).

מצאנו כי מספר הנוירונים האופטימאלי הוא 2. לאחר קיבוע ערך זה קיבל את אחוז הדיק הבאים -

test	train
0.7386	0.7894

2.3 קונפיגורציות

קונפיגורציה 1 – כוונון ערך weight decay

מנסה לכזב את המשקלים בראשת, במטרה למנוע מצב של over fitting. זהו סוג של "קנס" על משקלים גבוהים – משקלים גבוהים מדי יכולים ליצור התאמת יתר לניטוי האימון. נרצה למצוא בצורה איטרטיבית את הערך שיפחית את המשקלים במטרה להגדיל את אחוזי הדיק של סט האימונות. נרצה למצוא את הערך המיטבי עבור פרמטר זה שיגדל את אחוזי הדיק על סט האימונות (נבדוק זאת שוב בעזרת שיטת K-fold).

נתוני הקונפיגורציה: 30 נירונים בשכבה הכניסה, 2 נירונים בשכבה החוביה (ערך מקסימלי שנמצא בסעיף הקודם), מספר האיטרציות הוא 100, ערך decay האופטימאלי – 0.85 התקבל על ידי מקסום דיק סט האימונות (0.7644). ערך זה נמצא באמצעות אלגוריתם איטרטיבי שבחן את ערכי הפרמטר בקפיצות של 0.05 ([נספח 7](#)). כמות המשקלות הנלמדות – 52 ([נספח 8](#)).

קונפיגורציה 2 – הגבלת מספר איטרציות

מטרת קונפיגורציה זו היא מניעת מצב של over fitting. מציאת מספר האיטרציות שמספק את אחוזי הדיק הטובים ביותר עבור סט האימונות, כאשר בדיקה זו תבוצע בעזרת שיטת fold K. חסרון של קונפיגורציה זו עשוי להיות התאמת חסר בשל הורדת מספר האיטרציות, אך כדי להתמודד נבדוק מספר רב של אפשרויות ונבחר בטובה ביותר.

נתוני הקונפיגורציה: 30 נירונים בשכבה הכניסה, 2 נירונים בשכבה החוביה (ערך מקסימלי שנמצא בסעיף הקודם), מספר האיטרציות המיטבי הנמצא הוא 40 אשר מקסם את דיק סט האימונות (0.7674). ערך זה נמצא באמצעות אלגוריתם איטרטיבי שבחן את ערכי הפרמטר בקפיצות של 10 בתוווח הערכים 10-300 ([נספח 9](#)). כמות המשקלות הנלמדות – 52 ([נספח 8](#)).

סיכום קונפיגורציות

בכדי למצאו את ערכי הפרמטרים האופטימליים עשינו שימוש בשיטת fold K לחלוקת לסט אימון וסט אימונות. בכל לולאה חיצונית, הרצנו בלולאה פנימית את כל האופציות האפשרות לסט אימון ובחינה לפי שיטת fold K ואחוזי הדיק חושבו באמצעות ממוצע של כל אחוזי הדיק שחושבו בכל אופציה. בסוף, הערך האופטימאלי נבחר לפי אחוז הדיק הממוצע המקסימלי של סט האימונות.

תוצאות הדיק (לפי סט האימון המלא):

test	train	
0.7606	0.7832	Weight decay
0.7313	0.7669	Maxit limit

ניתן לראות כי אין הבדל מהותי בין שתי הקונפיגורציות. נבחר במודל שמקסם את אחוזי הדיק של סט הבדיקה, מכיוון שאחוזי דיק אלו משקפים כמה המודל טוב בחיזוי נתונים חדשים אשר לא התאמן עליהם. במקרה זה, נבחר בקונפיגורציה 1 אשר סיפקה אחוזי דיק גבוהים יותר. בנוסף, ביצענו שילוב של הקונפיגורציות, בחרנו לקבוע ערך אופטימאלי של פרמטר ולמצוא את ערכו האופטימאלי של הפרמטר השני בצורה איטרטיבית. ביצענו זאת פעמיים – עבור כל פרמטר שבחנו בקונפיגורציות הקודמות. ([נספח 10](#), [נספח 11](#))

test	train	הסבר	test	train	הסבר	
0.7768	0.7811	קיבענו את הערך האופטימאלי	Maxit = 40	0.7621	0.7832	קיבענו את הערך האופטימאלי Decay= 0.85 Opt Maxit = 150

		שנמצא עבור Maxit פרמטר	Opt decay= 0.6		שנמצא עבור decay פרמטר	
--	--	------------------------	----------------	--	------------------------	--

רצינו לבדוק האם כוונון של שני הפרמטרים ביחד, יספק תוצאות טובות יותר מאשר כוונון פרמטר יחיד בלבד. ניתן לראות שהשילוב בין הkonfiguraciot מושפר כמעט את אחוז הדיק ביחסו להשוואה למודלים בהם ביצעו כוונון פרמטר יחיד. לכן, נרצה לבחור בין המודלים שלקחו בחשבון את השילוב. ניתן לראות שעבור סט האימון כמעט אין הבדל בין המודלים, שכן נרצה לבחור לפי מודל שספק אחוז דיק גבוהים לסט הבדיקה ואיזון טוב יותר בין אחוז הדיק של הבדיקה והאימון, ככלומר נבחר במודול בו מספר האיטרציות הוא 40 וערךdecay האופטימאלי המתאים הוא 0.6. אמן, ניתן לראות כי ההבדל בין המודלים קטן מאוד, וכך להיות במצב שבו עבור סט הבדיקה אחר, נקבל אחוז דיק שוניים.

אילו נדרשנו לבחור פרמטר נוסף לכוונון, הינו בוחרים לשЛОוט במספר השכבות החבויות. לפי ברירת המחדל של חבילת `neuralnet` מסהך השכבות החבויות הוא 1. היכולת לאפשר מספר גדול יותר של שכבות חבויות יכול לאפשר למודל להיות מורכב יותר ומדויק יותר, אשר לומד את הנתונים בצורה מעמיקה. במקרה של שЛОוט על מספר השכבות החבויות, יש להשתמש בחבילת `neuralnet`, ונמצא את הפרמטר `hidden` אשר מזקם את אחוז הדיק של סט האימון.

3 – עצי החלטה Decision Trees

3.1 בניית עץ החלטה באמצעות סט אימון ואיומות

בשביל לבנות את עץ ההחלטה, נשתמש בפקודת `(part)`. בפונקציה זו, שולחים את סט האימון המלא (בל' חילוקה לסט אימון וסט איומות), מכיוון שפונקציית `part` מבצעת ולדיצה ובוחרת את העץ המזקם את הדיק של הווילדייה. ככלומר, השיטה מבצעת cross-validation מאחוריו הקלים. בנוסף, הפונקציה יודעת להתמודד עם משתנים קטגוריאליים שאינם ממורכזים, וכן נשתמש בסט האימון המקורי, ללא משתני דמה ולא ביצוע פונקציית `(scale)`.

3.2 הרצת פקודות printcp()

בטבלה מוצגים מספר מאפיינים:

- Root node error – מהי השגיאה המתקבלת אם בוצע סיווג רנדומאלי. זהה להסתברות האפרוריות של ערך "ק" במשתנה `u`.
- ערך `cp` – פרמטר הענשה על סיעוף (פיזולים) של העץ.
- `nsplit` – מספר הפיזולים בעץ.
- `rel error` – אומד את השגיאה של סט האימון. הכפלת משתנה זה בroot node error תיתן את השגיאה של סט האימון.
- `xerror` – כמו `rel error` עבור סט האימונות.

ערך ברירת המחדל של פרמטר `cp` של פקודה זו הוא 0.01, בכך לא להגביל את העץ, שינינו ערך להיות 0, בכך למצוא פרמטר `cp` עבורו שגיאת סט האימונות מינימאלית.

ניתן לראות כי ערכי הפיזול עולים אך לא באופן סדר (עמודת `nsplit`). אלגוריתם זה שואף להקטין את שגיאת `error`, ולכן במידה ופיזול מסוים לא הניב שיפור בערך זה, הוא לא יוצג בטבלה. בפלט הטבלה יוצגו רק פיזולים אשר הניבו שיפור כלשהו בערך זה.

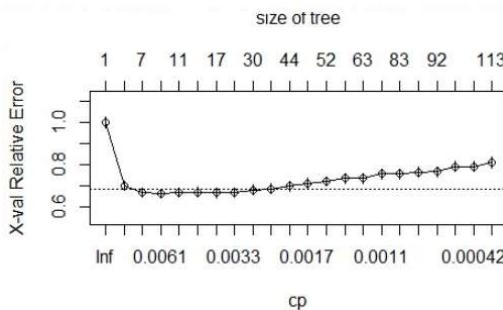
```
Classification tree:
rpart(formula = Train_Set_Tree$ ~ ., data = Train_Set_Tree,
      method = "class", parms = list(split = "information"), cp = 0)

Variables actually used in tree construction:
[1] age          balance    campaign   day        education   housing    job
[9] marital     month      pdays      poutcome   previous   loan

Root node error: 971/2725 = 0.35633
n= 2725
```

	CP	nsplit	rel error	xerror	xstd
1	0.15036045	0	1.00000	1.00000	0.025747
2	0.01510470	2	0.69928	0.69928	0.023253
3	0.00823893	6	0.63852	0.66735	0.022888
4	0.00446275	7	0.63028	0.66220	0.022827
5	0.00411946	10	0.61689	0.66735	0.022888
6	0.00377618	13	0.60453	0.66941	0.022912
7	0.00360453	16	0.59320	0.66632	0.022876
8	0.00308960	20	0.57878	0.66838	0.022900
9	0.00257467	29	0.54892	0.67971	0.023032
10	0.00205973	37	0.52832	0.68589	0.023103
11	0.00185376	43	0.51596	0.69825	0.023242
12	0.00154480	49	0.50360	0.71164	0.023389
13	0.00137315	51	0.50051	0.72091	0.023489
14	0.00128733	54	0.49640	0.73532	0.023640
15	0.00117699	62	0.48610	0.73532	0.023640
16	0.00102987	69	0.47786	0.75695	0.023860
17	0.00085822	82	0.46447	0.75592	0.023850
18	0.00068658	88	0.45932	0.76313	0.023921
19	0.00058849	91	0.45726	0.76725	0.023961
20	0.00051493	98	0.45314	0.79094	0.024187
21	0.00034329	106	0.44902	0.79197	0.024196
22	0.00000000	112	0.44696	0.81050	0.024365

ערך cp מינימלי המתקיים הוא 0.004463 (מסומן בטבלה וניתן לראות גם בגרף), נבחר לפי הערך המינימלי בעמודת `xerror`.

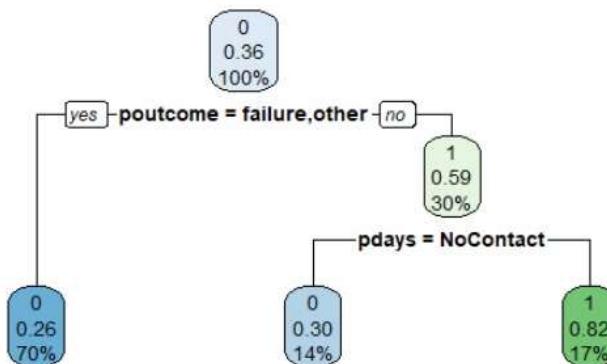


לפי ערך cp שהתקבל, נבצע קטימה של העץ.

test	train	
0.7386	0.8407	לפני קטימה
0.7709	0.7754	אחרי קטימה

ניתן לראות כי לפני הקטימה ישנו מצב של *over fitting*, המודל מתאים מאוד לסט האימון, אך יש פער משמעותי בין אחוזי הדיק בסט האימון לבין אחוזי הדיק בסט הבדיקה (בסט הבדיקה קיבלונו אחוזי דיק נמוכים יחסית) ([סעיף 12](#)). לעומת זאת, לאחר הקטימה, ניתן לראות כי לא קיים הבדל משמעותי בין אחוזי הדיק של סט האימון לבין אחוזי הדיק של סט הבדיקה ([סעיף 13](#)).

העץ המתקיים לאחר קטימה אחת לא ניתן להציג בצורה ברורה ([סעיף 14](#)). אך נבצע קטימה נוספת לפי cp גבוה יותר בכך שונכל להציג את העץ בצורה ברורה.



רשומה לדוגמה מסט הבדיקה (נציג רק את הערכים בהם העץ עושה שימוש) –

Record num	education	housing	loan	quarter	pdays	poutcome
980	tertiary	yes	no	4	NoContact	failure

הຮסתעפות הראשונה בעץ היא לפי `poutcome`, בגל שהערך הוא `failure` נסתעף ימינה. לאחר מכן, הרסתעפות היא לפי `pdays`, הערך שלנו הוא `NoContact`, ולכן נסתעף שמאליה, והחיזוי יהיה 0 – כולם הלקוח לא צטרף לתוכניות הפיקדון. ערך הרשמה הוא גם 0, ככלומר החיזוי תואם את המציאות.

התובנות שניתנו להסביר מהו העץ הן לגבי חשיבות השונה של המאפיינים השונים בבעיה. מאפיינים שמקורם גבוה בעץ (קרוב לשורש) הם בעלי יכולת סיוג מיטבית מtower כל המאפיינים בבעיה. למשל, במקרה שלנו המסזג האיכותי ביותר הוא `poutcome`, מה שמתישב עם מתאם בין משתנה זה למשתנה המטרה כפי שראינו בחלק א' ([סעיף 15](#)).

לעומת זאת, משתנים שנקבעו עקב קביעת ערך cp ולא נכנסו לעץ, לא מספקים מידע מספק בכך לסייע את הרשומות, וכן לא ילקחו בחשבון בעת קבלת החלטת הסיוג.

3.3 קטימת העז לפि ערכי Cp נוספים

test	train	
0.7386	0.8407	Cp = 0
0.6755	0.6437	Cp (max xerror) = 0.15036
0.7665	0.7846	Cp (second best xerror) = 0.00412
0.7783	0.7508	Cp (2 splits) = 0.015105

עבור $C_p = 0$ קיבל את אותו העז לפני הקטימה כפי שקיבלנו בסעיף הקודם. במצב זה ניתן להזות מצב של over fitting, כי אחוזי הדיק על סט האימון גבוהים במיוחד, ולעומת זאת, אחוזי הדיק של סט הבדיקה נמוכים משמעותית.

עבור ערך הקפ שהוביל לשגיאה הגבוהה ביותר על סט הולידציה ניתן לראות כי הדיק על סט הבדיקה גבוהה מהתוצאות על סט האימון. ערך C_p זה הוא הערך הגבוה ביותר שנמצא בטבלה, כלומר הקטימה מתבצעת בשלב מאוד מוקדם ולן העז לא מספיק מרכיב בצד לסייע בצורה טובה את מאפייני הבניה (מצב של under fitting).

את הקפ השלישי בחרנו בתור זה שנותן את הערך השני הכי נמוך של x_{error} , מהסיבה x_{error} עולה במעט יחסית ($\text{עליה של } 0.005$) ועם זאת קיבל שיפור יחסית יותר משמעותי בערך $x_{\text{error rel}}$ (קטן ב-0.02). במצב זה, קיבלנו ש אחוזי הדיק בין סט האימון לבדינה יחסית מאזורים, אם כי הדיק על סט האימון גבוהה יותר.

בחרנו לבחון ערך C_p רביעי, אשר מתאר מודל עז עם 2 פיצולים, אשר מפשט את המודל בצורה משמעותית. במקרה זה, הלמידה מתבצעת על פי 2 מאפיינים בלבד. ניתן לראות שעבור ערך C_p זה, מתקבלים אחוזי דיק הדומים מאוד לערכי הדיק עבור C_p האופטימאלי. ההפרש בין אחוזי הדיק של סט האימון וסט הבדיקה לא גדולים במיוחד, אם כי אחוזי הדיק עבור סט האימון נמוכים יותר מאשר של סט הבדיקה.

3.4 השוואת חמישת המודלים

נרכז את מאפייני העצים השונים שקיבלנו בסעיפים הקודמים:

train - test	xerror	splits	test	train	CP
0.45	0.6622	7	77.09	77.54	$C_p \text{ Opt} = 0.004463$
10.21	0.8105	112	73.86	84.07	$C_p = 0$
- 3.18	1	0	67.55	64.37	$C_p (\text{max } x_{\text{error}}) = 0.15036$
1.81	0.6674	10	76.65	78.46	$C_p (\text{second best } x_{\text{error}}) = 0.00412$
- 2.75	0.6993	2	77.83	75.08	$C_p (2 \text{ splits}) = 0.015105$

קriterוניים לבחירת המודל:

- **אחוזי הדיק על סט הבדיקה – בקריטריון זה, נרצה לבחור את הערך המקסימלי, שמתאר יכולת ניבוי טובה לרשומות שהמודל לא התאים עליהן. ערך זה בעצם מתאר את מבחן התוצאה, ואת אחוזי הדיק שנתקבל ביצוע תחזיות חדשות.**
- **מספר פיצולים – נרצה לבחור במודל שבו מספר הפיצולים לא גדול במיוחד, מכיוון שמספר גדול של פיצולים מסביר את המודל ומאריך את זמן ביצוע התחזית, וכן נרצה מספר לא גדול של פיצולים כדי לפשט את המודל.**
- **הפרשים בין אחוזי דיק של האימון לבין הבדיקה – ההפרש באחוזי הדיק בין סט האימון לסט הבדיקה יכול להעיד על מצב של התאמת יתר, נרצה לבחור בסט המאוזן ביותר שבו ההפרש הוא הקטן ביותר.**
- **xerror – נרצה לבחור את הערך הקטן ביותר, ככלומר את השגיאה הקטנה ביותר על סט האימונות.**

לאחר שכלל הקriterוניים השונים, בחרנו בערך C_p האופטימאלי = 0.004463, עבורו מתקבל האיזון הטוב ביותר בינו לדיק סט האימון לסט הבדיקה, גם מבחינת אחוזי דיק בסט המבחן וסט הולידציה מתקבלים אחוזים גבוהים ביחס לשאר, ומספר הפיצולים בערך עדין נמוך יחסית ולא מסרב את המודל.

K - Means 4

4.1 הוצאת משתנה המטרה מתוך ה

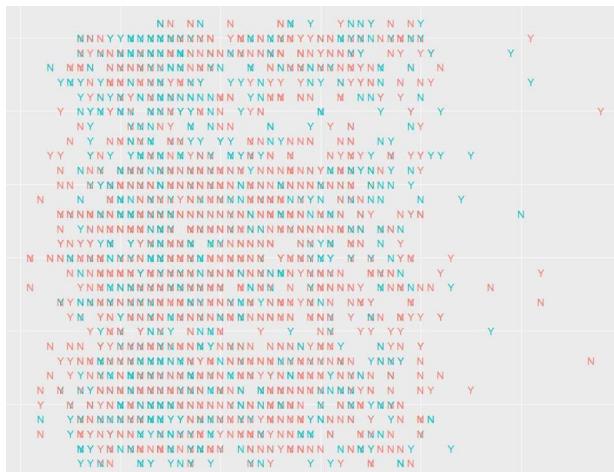
אלגוריתם k-means הוא שיטה לניטוח אשכולות (clustering), אשר מבצע למידה לא מונחת, כלומר בקהלט נכניס את מספר המחלקות הקיימות אך לא את החלוקה למחלקות בפועל. לכן, נרצה להוציא את המשתנה המטרה מהקהלט בכך לאפשר את הלמידה הלא מונחת.

4.2 הרצת מודל עם ערכי ברירת מחדל

לפני הרצת המודל, נצטרך להכין את הנתונים להרצת k-means. את המאפיינים הקטגוריאליים נציג בעזרת משתני דמה (כפי שעשינו בرشת הנירונים), ונ裏ץ את הפקודה (`scale` בצד ימין וליצור התאמה בסדרי הגודל של המשתנים).

ביצענו הרצת המודל עם ערכי ברירת מחדל, ועם $k=2$ כי בבעיה שלנו קיימות 2 מחלקות בלבד – האם הלוקוט הצטראף לתוכנית פיקדון (1) או לא (0).

בצד ימין חשב את אחוזי הדיק של האימון, ביצענו השוואה בין הקבוצות לבין הסיווג לקבוצות בפועל (0\1). מבדיקה עברו איזו מחלוקת מקבילה לתשובה "כן" ואיזו מחלוקת מתאימה לתשובה "לא" כביכול, בדקנו את שתי האפשרויות ובחרנו באפשרות הסבירה יותר שמצבעת התאמה באופן טוב יותר. ההתאמה שהתקבלה היא שסיווג לקבוצה 2 מתקבל על ערך 1 – כלומר מצטראף לתוכנית פיקדון, ולהיפך. אחוז הדיק לחושב לפי מספר הרשומות שהסיווג שלון תואם למציאות חלק סה"כ הלוקחות. בנוסף, הצגנו גרפ' שמתארו להראות לאיזה קבוצה סוג הלוקוט לפי ע (אדום – "לא", ירוק – "כן") אל מול הערך הנצפה שמתואר על ידי אותן (N – "לא", Y – "כן"). ניתן לראות בגרף את 2 סוג הטעויות (תrzפיות N שסווgo כירוק ולהיפך).



אחוז הדיק של סט האימון חושבו לפי $\frac{\sum(y=clu_)}{number\ of\ rows}$. כלומר, 61.47% מחלקות סוגו בהתאם לקבוצה בפועל.

4.3 טיב ההתאמה בין האשכולות למחלקות

במטרה לבדוק את טיב ההתאמה בין האשכולות למחלקות, ראיינו כי אחוז הדיק של האימון אינם גבוהים במיוחד. כאשר בוחנים אלגוריתם אשכול, נרצה להגיע למסוב שבו המרחקים בין המחלקות השונות הם הגודלים ביותר, ולעומת זאת, בתוך המחלוקת – קטנים ביותר. שני אלו מתקשרים לשני מושגים בהקשר של סיווג הקבוצות – *betweeness* – *wittiness*. נרצה לבחון 2 מדדים אשר לוקחים בחשבון את שני מרחקים אלו, כאשר המטרה שלנו היא למדוע את DB (davis bouldin) המתיחס למרחק בתחום המחלוקת, ולמנסם את חחון המתיחס למרחק בין המחלוקות.

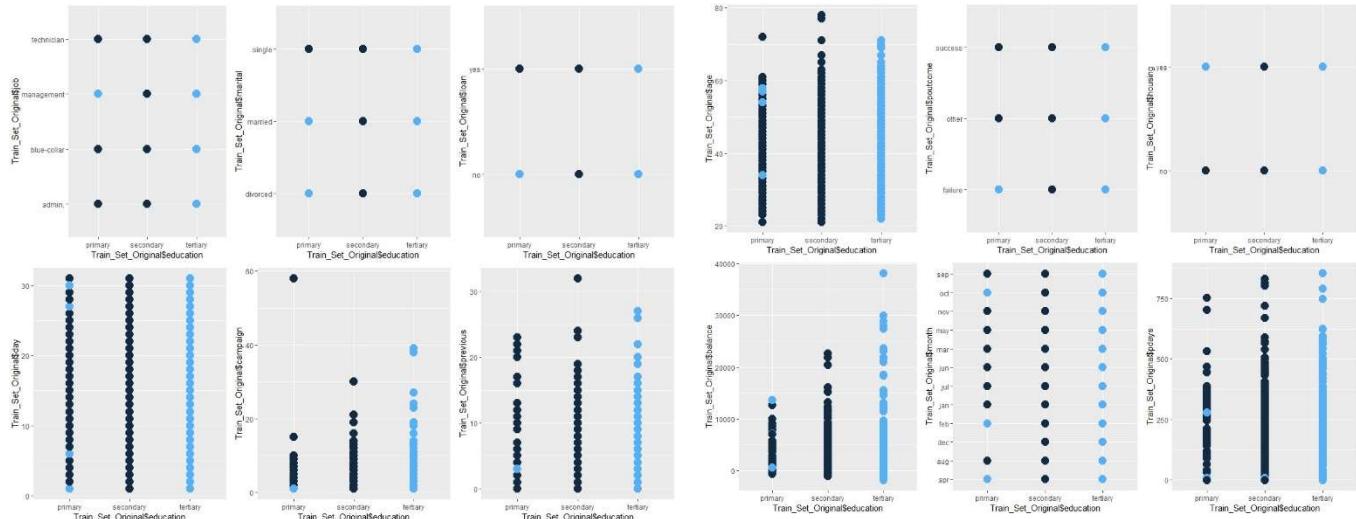
לאחר הרצת אלגוריתם k-means על סט האימון, קיבלנו את המדדים הבאים:

Value (train)	parameter
0.7103	dunn
2.745	DB

בנוסף, רצינו לבדוק מאפיינים חסודים היכולים להשפיע על האשכול. בדקנו את כל המאפיינים בסט הנתונים ([16](#)), ובחרנו להציג את אלו שהפיקו את החלוקה המעניינת ביותר.

מאפיין education בתור מסוכן

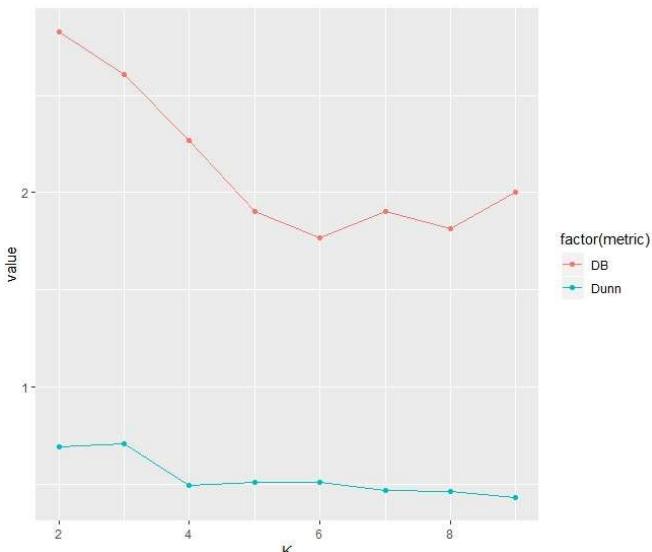
בהתכלות על כל המאפיינים, ניתן לראות כי לפי משתנה זה, ישנה חלוקה יחסית ברורה למחקות השונות. ככלומר, אפשר להגיד שמשתנה זה מראה מטוגן טוב עבור הנתונים בהקשר של אשכול באמצעות K-means (תכלת - סיווג למחקה 1, כחול - סיווג למחקה 2). בחלק הקודם של העבודה, רأינו כי רמת השכלה שונה משפיעה באופן שונה



על משתנה המטרה, ולכן תוצאות אלו מתישבות עם ההשערות שלנו.

4.4 ערך A מיטבי

בדדי נמצא ערך A מיטבי (כלומר מספר המחלקות המיטבי לשט הנתון), נבצע אימון על ערכי A שונים. עבור כל A נציג 2 מדדים – DB והערך DB – Dunn. מעריכים אלו מתראים את טיב המודל כפי שפורט מעלה. ערך A אופטימאלי יבחר לפי ערכי המדדים, כאשר ברכזונו למקסם את חחנס (המתיחס למרחק בין המחלקות) ולמצער את DB (המתיחס למרחק בתוך המחלקה). בחנו 8 ערכי A, מ2 ועד 9.



לפי הגרף, ניתן לראות כי נקודת המינימום של DB ונקודת המינימום של חחנס לא מתקבלות עבור אותו k . לכן, נרצה לבחור ערך אופטימלי לפי ערך k שמספק את המרחק המינימלי בין מדדים אלו. לפי זה, ניתן לראות כי הערך האופטימלי הוא 6. הערך גבוה יותר מrangle מחלוקת הקיימות בסיפור המקירה (2 מחלוקת – "כן" לא"). היה ומדובר בלקוחות שהם בני אדם שונים במאפיינים רבים, אפשר להסביר את החלוקה במספר רב יותר של מחלוקות. ככלمر הלקוחות לא נבדלים רק בהם יסכנו או לא יסכנו לתוכנית הפיקדון, אלא אפשר לסוג אותם במספר רב יותר של אשכולות לפי תכונות נוספות.

K	DB	dunn
2	2.827898	0.6923256
3	2.607267	0.7048045
4	2.269603	0.4952623
5	1.901769	0.5111336
6	1.764792	0.5102599
7	1.900756	0.4688037
8	1.813299	0.4626341
9	2.002027	0.4305098

SVM 5

בחרנו לבדוק מודל נוסף – מודל SVM (מכונות וקטוריים תומכים), שנלמד בקורס בכיתה. זהו טכניקה של למידה מונחית היכלה לשמש לניטוח נתונים, סיווג ורגסיה. בחרנו להשתמש בטכניקה זו על מנת לסוג את הנתונים ולבצע תחזיות למשתנה המוסבר שלנו. מטרת המודל היא למצוא עיקום הפרדה אשר יוצר מרוחה גדולה ככל הנתון בין לבני התכיפות הקורבות לו ביוטר משתי המחלוקות. הרצת המודל נעשו בעזרת פונקציית `svm` מתוך חבילת [e1071](#) ([נוף](#)).

לאחר הרצת המודל קיבלנו את תוצאות הדיק הבות –

test	train
0.7753	0.7824

6 השוואת המודלים

במהלך העבודה בדקנו 4 מודלים. נרצה למצוא דרך להשוות בין ביצועי המודלים השונים בכך לבחור את המודל הטוב ביותר. שלושה מודלים הם מונחים – רשת נירוניים, עץ החלטה, ומודל SVM, ומודל הנוסף הוא מודל לא מונחה – אלגוריתם `k-means`. אלגוריתם `k-means` מבצע למידה איטרטיבית על סט האימון ובמצע חלוקה לקבוצות. בכך ליהوت מסוגלים להשתמש באלגוריתם לביצוע תחזיות על סט הבדיקה, יש להשתמש במרכזי המחלוקות שנלמדו בסט האימון, ולבחן עבור הנתונים החדשניים – לאיזה מחלוקת הם משווים לפוי קרבה למחלוקות השונות. לאחר שתהיה בידנו פונקציית `chisq` שימושת במרכזיים שנמצאו באלגוריתם `k-means`, נוכל לחשב לשיטה זו את הדיק על סט הבדיקה, ואז נוכל להשוות בין המודלים השונים על ידי אחוז הדיק על סט הבדיקה. כאשר החלטה שלנו תתקבל על ידי בחירת המודל עבורו אחוז הדיק על סט הבדיקה הוא הטוב ביותר. השתמשנו בפונקציית `predict` של חבילת `ccclust`, אשר עשוה שימוש במרכזי המחלוקות ומסוגת כל תצפית למחלוקת הדומה לה ביותר. אנחנו בחרנו להשוות את איכות הסיווג של `k-means` בחלוקת ל-2 מחלוקות ובדיקת החלוקת על נתונים חדשים שלא התאימו

עליהם, ובכך לבדוק את איכות הסיווג שלו וההתאמה למחקרים השונות. אם הינו רצים לשימוש ב- k המיטבי (6) שמצאו בעזרת אינדקסים DB ו-*churn*, הינו צריכים לחשב על דרך אחרת- אולי להכניס את מספר המחלקה אליה סיווג התצפית ממשתנה מסביר נוסף לאחד מהמודלים המונחים. אך כאמור, בחרנו להשאיר את k להיות 2 ולהתאים למסגרת הסיפור ובכך להשוות בין המודלים. להלן סיכום חמשת המודלים:

Test	Chosen Model	
0.7768	Maxit = 40 Opt decay= 0.6	Neural Networks
0.7709	Cp Opt = 0.004463	Decision Tree
0.5389	k = 2	K - means
0.7753		SVM

ניתן לראות כי המודלים המונחים מספקים אחוז דיק טובי יותר על סט הבדיקה מאשר אלגוריתם *means-k*. הסיבה לכך היא ששאנס k לא בהכרח מסוויג לאשכולות לפי משתנה המטריה, ולכן לא תמיד מתאפשרת ההתאמה בין התחזית לבין ערך המשתנה המטריה של התצפית בפועל. בין 2 המודלים המונחים, לפי אחוז הדיק על סט הבדיקה, נבחר ברשת נוירונית. אם כי, ההפרש בין אחוז הדיק אינם גבוהים משמעותית מאשר עץ החלטה שקיבלנו. בנוסף, ראיינו קודם כי רשת הנוירונית שבחרנו מדוינת, כלומר ההפרש בין אחוז הדיק על סט האימון לבין אחוז הדיק על סט הבדיקה מזערי וכן אין חשש ל*over fitting*.

השוואה בין המודלים השונים -

סוג הלמידה	מודלים	ריצויים	מבנה	יתרונות	חסכנות
מנוחית	לא מנוחית	ביצועים יחסית טובים על סט הנתונים.	מודל יחסית פשוט. סיבוכיות זמן הריצה תנגד באופו מעריצי ביחס לגודל הבעיה. לעיתים נדרש קיטימת העץ לミニעת ההתאמת יתר. הקיטימה מתבצעת באמצעות קביעת ערכו של פרמטר CP.	ברשת נוירונים עושים שימוש בפונקציות אקטיבציה שונות. יש אפשרות להשתמש בשינוי מספר האיטרציות במהלך ריצה, מספר הנוירונים בשכבה החובית וכו'. כל אלו מօספים פרמטרים אשר משפיעים על מרכיבות המודל. אלגוריתם זה בעל סיבוכיות זמן ריצה גבוהה.	בפונקציות אקטיבציה שונות. יש אפשרות לשנות שינוי מספר האיטרציות במהלך ריצה, מספר הנוירונים בשכבה החובית וכו'. כל אלו מօספים פרמטרים אשר משפיעים על מרכיבות המודל. אלגוריתם זה בעל סיבוכיות זמן ריצה גבוהה.
ביצועים	ביצועים יחסית טובים על סט הנתונים.	מודל יחסית פשוט. סיבוכיות זמן הריצה תנגד באופו מעריצי ביחס לגודל הבעיה. לעיתים נדרש קיטימת העץ לミニעת ההתאמת יתר. הקיטימה מתבצעת באמצעות קביעת ערכו של פרמטר CP.	מודל פשוט להבנה ולקבלת אינטואיציה על הנתונים. בדרך כלל הגיוני ואפשר להבין את מבנה העץ. ניתן להבין מהעץ מה המאפיינים בעלי חשיבות גבוהה ביותר.	הרש ת יכולה להציג לרמת סיבוכיות גבוהה וכן גם יכולת הדיק שללה تعالיה בהתאם. אפשרות ניתוח קשרים לא ליאරיים. בעלת יכולת לשמש כמנגנון קירוב פונקציה שרירותי שלומד מנתונים שנצפו. דינמיות במבנה הרשת.	מודל פשוט להבנה ולקבלת אינטואיציה על הנתונים. בדרך כלל הגיוני ואפשר להבין את מבנה העץ. ניתן להבין מהעץ מה המאפיינים בעלי חשיבות גבוהה ביותר.
מבנה	לא מנוחית	מודל פשוט להבנה ולקבלת אינטואיציה על הנתונים.	מודל פשוט להבנה ולקבלת אינטואיציה על הנתונים. בדרך כלל הגיוני ואפשר להבין את מבנה העץ. ניתן להבין מהעץ מה המאפיינים בעלי חשיבות גבוהה ביותר.	הרש ת יכולה להציג לרמת סיבוכיות גבוהה וכן גם יכולת הדיק שללה تعالיה בהתאם. אפשרות ניתוח קשרים לא ליאריים. בעלת יכולת לשמש כמנגנון קירוב פונקציה שרירותי שלומד מנתונים שנצפו. דינמיות במבנה הרשת.	מודל פשוט להבנה ולקבלת אינטואיציה על הנתונים. בדרך כלל הגיוני ואפשר להבין את מבנה העץ. ניתן להבין מהעץ מה המאפיינים בעלי חשיבות גבוהה ביותר.
חסכנות	חסור שקייפות - לא ניתן לראות את הריצום שמאחורי ההחלטה של המודל. זמן ריצה ארוך יחסית.	בסביבות גובהה יותר מאשר מודלים אחרים יכולה לזרום למצב	חישוב שקייפות - לא ניתן לראות את הריצום שמאחורי ההחלטה של המודל. זמן ריצה ארוך יחסית.	חישוב שקייפות - לא ניתן לראות את הריצום שמאחורי ההחלטה של המודל. זמן ריצה ארוך יחסית.	חישוב שקייפות - לא ניתן לראות את הריצום שמאחורי ההחלטה של המודל. זמן ריצה ארוך יחסית.

<p>ניתקל בקושי בהערכת מספר קבוצות שקיבלו/algoritm. איננו מתמודד טוב עם תוצאות בעלות פיזור גבוה. במידה ותוצאות הדיקט גבואה, עדין קשה להבין מהם המאפיינים בעלי הקשר הטוב ביותר למשתנה המטרה.</p>	<p>נטיה להסתאמת יתר. זמן החישוב גדל מעריכית ביחס לגודל הבעיה. בנוסף, ניתן לשינויים על סט האימון – שינוי קל עשוי ליצור עז שונה.</p>	<p>של fitting over. רנדומליות המשקלים מחדירה תוצאות שונות ולכך קשה להשוות בין המודלים המתקבלים. לעיתים נגיעה לכך' קיצון מקומית ולא לנק' קיצון הגלובאלית ולכן הפתרון לא יהיה אופטימאלי.</p>
--	--	--

7 המודל הנבחר

7.1 הצגת המודל הנבחר

כפי שתכננו, בחרנו במודול של רשת נוירונים. המודול שבחרנו מורכב משכבה חבויה אחת שבה 2 נוירונים כשבכמת הכניסה 2 נוירונים בשכבת היציאה. במודול עשינו שימוש ב40 איטרציות וערך 0.6 עבור פרמטר decay. מודל זה סיפק את הביצועים הטובים ביותר מבין המודלים השונים שנבחרו בראשת הנוירונים, ואת אחוז הדיקט הטובים ביותר על סט הבדיקה בהשוואה למודלים הנבחרים השונים (ע"ז החלטה, SVM ו-k means). מאפייני הקונפיגורציה האלה נבחרו מכיוון שהם סיפקו את התוצאות הטובות ביותר בשלב ההשוואה בין רשתות הנוירונים השונות. בחרנו לעשות שילוב בין שניי של 2 פרמטרים שונים יחד, כפי שתואר בפרק רשת נוירונים ([קישור להשוואת רשתות הנוירונים](#)).

7.2 ניתוח תוצאות בעזרת מטריצת מבוכה

מטריצת המבוכה מציגה את התוצאות בפילוח לפי הערך האמיתי של התצפית. באמצעותה ניתן לראות טיעויות מסוגים, ואת הצלחות החיזוי.

		true_values	
		prediction	0 1
prediction	0	414	106
	1	46	115

במקרה שלנו, ניתן לראות כי ב-90% נזהה באופן נכון לקו שאים מעוניין להצטרף $= \frac{414}{414+46} = \frac{414}{460}$, ו-52% לקוות אשר אכן מעוניינים להצטרף לתוכנית $= \frac{115}{115+106} = \frac{115}{221}$. ניתן לראות כי קיימת הטיה של המודול לערך תחזית 0 – "לא", זה מתיישב עם ההסתברות האפריאורית של משתנה המטרה אשר גם היא מוגה לפ"י ערך זה.

8 הגשת חיזויים סופיים

בשביל לבצע חיזויים סופיים ולהשתמש בראשת הנוירונים שבחרנו, יש צורך בביצוע מספר התאמות של סט הנתונים כך שיוכלו להיכנס לרשף, וביצוע התמורות למאפיינים השונים כפי שעשינו בתחלת התהילה. למשל, הורדנו את אוטם המשתנים שבחרנו להוריד – gender, default, contact – לאחר הכתת סט הנתונים, יכלנו לבצע את התחזיות כנדרש.

9 נספחים

9.1 הנקה הנתונים נספח 1 – עיקרי חלק א'

- הורדת 3 משתנים – gender, default, contact
- הפיכת משתנה month, pdays למשתנים קטגוריאליים.
- השלמת נתונים חסרים בעזרת אלגוריתם mice.

נספח 2 – חלוקה לסט אימון ובחינה

```
# divide to train and test sets

set.seed(123)
Rows_Numbers_for_Test_Set <- sample(1:nrow(Bank_Data), 0.2*nrow(Bank_Data), replace = FALSE)

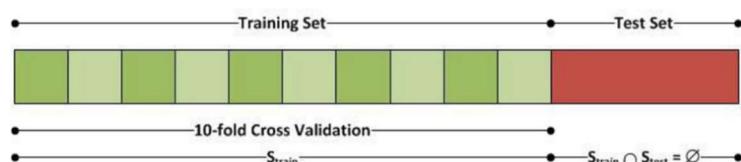
Train_Set <- Bank_Data[-Rows_Numbers_for_Test_Set,]
Test_Set <- Bank_Data[Rows_Numbers_for_Test_Set,]
```

נספח 3 – חלוקה לסט אימון ואמונות

את סט הנתונים נדרשנו לחלק לשולשה חלקים – סט אימון (training), סט אימונות (validation) ובחינה (test). את החלוקה ביצענו בשלבים – קודם כל ביצענו חלוקה לסט אימון training וסט בחינה test. בחרנו להקצות 20% מסט הנתונים לסט בחינה (681 נתונים), אחרי סקירה לגבי החלוקה הנוהגה לסדר גודל נתונים כמו שלנו. בכך לאمن את המודל כפי שציר, רצינו להקצות את מירב הנתונים לסט האימון (80%) בכך לאפשר למודל להיחשף לרשותות שונות ומגוונות.



לאחר מכן, יש לחלק את סט האימון לסט אימון וסט אימונות. את החלוקה נבצע באמצעות שיטת *k-fold cross validation*, הנקראת *k-fold*, אשר מבצעת חלוקה של סט האימון כולה ל-*k* חלקים, ובכל איטרציה מקצים 1-*k* מהקבוצות לסט האימון ואת הקבוצה הנոתרת לסט אימונות. שיטה זו הינה השיטה הפופולרית ביותר מ בין שיטות cross validation ומעניקה בחינת כל אחת מההypotheses גם בתור אימון, וגם בתור אימונות, ובוחרת את סט הנתונים שמאפיין את נתוני החסובים ביותר. בחרנו להשתמש ב-*k*=5, לפי ערך זה, מתקבל כי מתוך 80% מההווים 2725 נתונים, כ- 20% (בגלל החלוקה הרנדומאלית לקבוצות) המהווים כ- 545 נתונים יהיו סט האימונות, וכל השאר, כ- 2180, יהיו סט האימון.



עשינו שימוש בשיטת *k-fold* עם *k*=5. כלומר, חילקנו את סט האימון ל-5 קבוצות בצורה רנדומאלית, והרכבנו 5 סטים שונים כאשר בכל סט קבוצה אחרת מהווה את סט הויידציה, והשאר – 4 הקבוצות האחרות, מהוות את סט האימון.

```

# K-fold configuration
n.folds <- 5
samples <- sample(1:n.folds, nrow(train_set), replace=T)
Train_and_val <- cbind(train_set, samples)
k_fold_sets <- list()
for(fold in 1:n.folds){
  k_fold_sets[[fold]] <- Train_and_val[Train_and_val$samples==fold,1:ncol(Train_and_val)-1]
}

k_folds_train_sets <- list()
k_folds_valid_sets <- list()

k_folds_train_sets[[1]] <- rbind(k_fold_sets[[2]],k_fold_sets[[3]],k_fold_sets[[4]],k_fold_sets[[5]])
k_folds_train_sets[[2]] <- rbind(k_fold_sets[[1]],k_fold_sets[[3]],k_fold_sets[[4]],k_fold_sets[[5]])
k_folds_train_sets[[3]] <- rbind(k_fold_sets[[1]],k_fold_sets[[2]],k_fold_sets[[4]],k_fold_sets[[5]])
k_folds_train_sets[[4]] <- rbind(k_fold_sets[[1]],k_fold_sets[[2]],k_fold_sets[[3]],k_fold_sets[[5]])
k_folds_train_sets[[5]] <- rbind(k_fold_sets[[1]],k_fold_sets[[2]],k_fold_sets[[3]],k_fold_sets[[4]])

k_folds_valid_sets[[1]] <- k_fold_sets[[1]]
k_folds_valid_sets[[2]] <- k_fold_sets[[2]]
k_folds_valid_sets[[3]] <- k_fold_sets[[3]]
k_folds_valid_sets[[4]] <- k_fold_sets[[4]]
k_folds_valid_sets[[5]] <- k_fold_sets[[5]]

```

נסוף 4 – המרת משתנים קטגוריאליים למשתני דמה

#משתנה pdays

רמת הבסיס היא – NoContact. הגדרנו בעזרת 3 משתני דמה את 4 הקטגוריות

$$pdays_D1 = \begin{cases} 1 & \text{HalfYear} \\ -1 & \text{else} \end{cases} ; pdays_D2 = \begin{cases} 1 & \text{OneYear} \\ -1 & \text{else} \end{cases} ; pdays_D3 = \begin{cases} 1 & \text{OverOneYear} \\ -1 & \text{else} \end{cases}$$

#משתנה job

רמת הבסיס היא – admin. הגדרנו בעזרת 3 משתני דמה את 4 הקטגוריות

$$job_D1 = \begin{cases} 1 & \text{blue-collar} \\ -1 & \text{else} \end{cases} ; job_D2 = \begin{cases} 1 & \text{management} \\ -1 & \text{else} \end{cases} ; job_D3 = \begin{cases} 1 & \text{technician} \\ -1 & \text{else} \end{cases}$$

#משתנה marital

רמת הבסיס היא – single. הגדרנו בעזרת 2 משתני דמה את 3 הקטגוריות

$$marital_D2 = \begin{cases} 1 & \text{divorced} \\ -1 & \text{else} \end{cases} ; marital_D1 = \begin{cases} 1 & \text{married} \\ -1 & \text{else} \end{cases}$$

#משתנה education

רמת הבסיס היא – primary. הגדרנו בעזרת 2 משתני דמה את 3 הקטגוריות

$$education_D2 = \begin{cases} 1 & \text{tertiary} \\ -1 & \text{else} \end{cases} ; education_D1 = \begin{cases} 1 & \text{secondary} \\ -1 & \text{else} \end{cases}$$

#משתנה quarter

רמת הבסיס היא – רביעון ראשון. הגדרנו בעזרת 3 משתני דמה את 4 הקטגוריות

$$quarter_D1 = \begin{cases} 1 & \text{second} \\ -1 & \text{else} \end{cases} ; quarter_D2 = \begin{cases} 1 & \text{third} \\ -1 & \text{else} \end{cases} ; quarter_D3 = \begin{cases} 1 & \text{fourth} \\ -1 & \text{else} \end{cases}$$

#משתנה poutcome

רמת הבסיס היא – success. הגדרנו בעדרת 2 משתני דמה את 3 הקטגוריות

$$poutcome_D1 = \begin{cases} 1 & \text{failure} \\ -1 & \text{else} \end{cases} ; poutcome_D2 = \begin{cases} 1 & \text{other} \\ -1 & \text{else} \end{cases}$$

משתנה loan

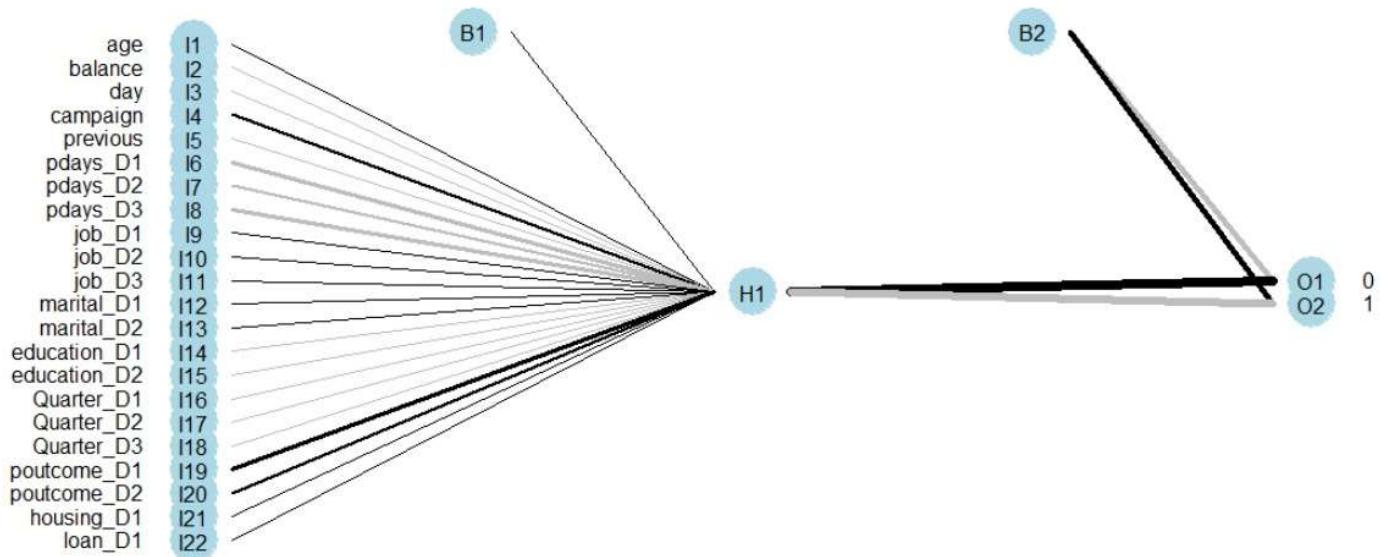
$$loan_D1 = \begin{cases} 1 & \text{yes} \\ -1 & \text{no} \end{cases}$$

משתנה housing

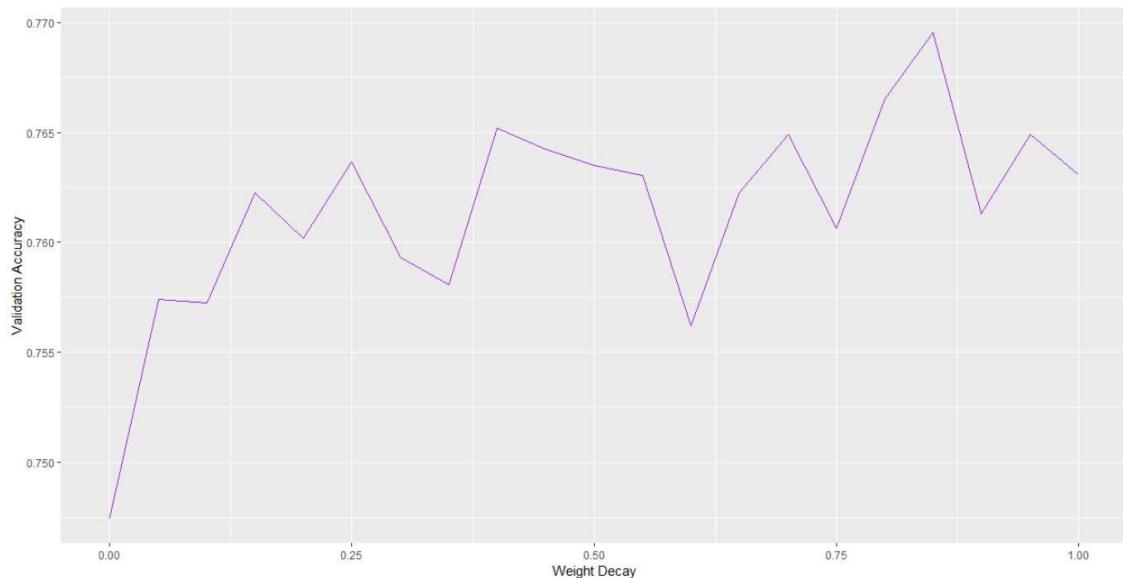
$$housing_D1 = \begin{cases} 1 & \text{yes} \\ -1 & \text{no} \end{cases}$$

9.2 רשת ניירונים

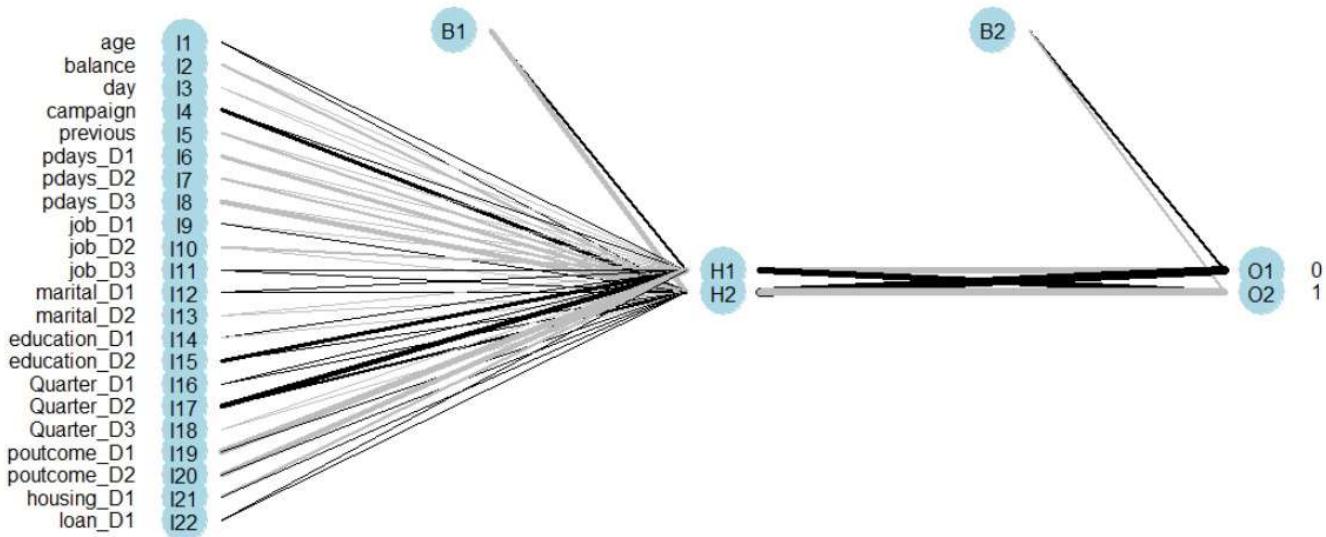
נספח 6 - הרצת הרשת עם נתוני ברירת מחדל



נספח 7 – ערך אופטימאלי קונפיגורציה decay

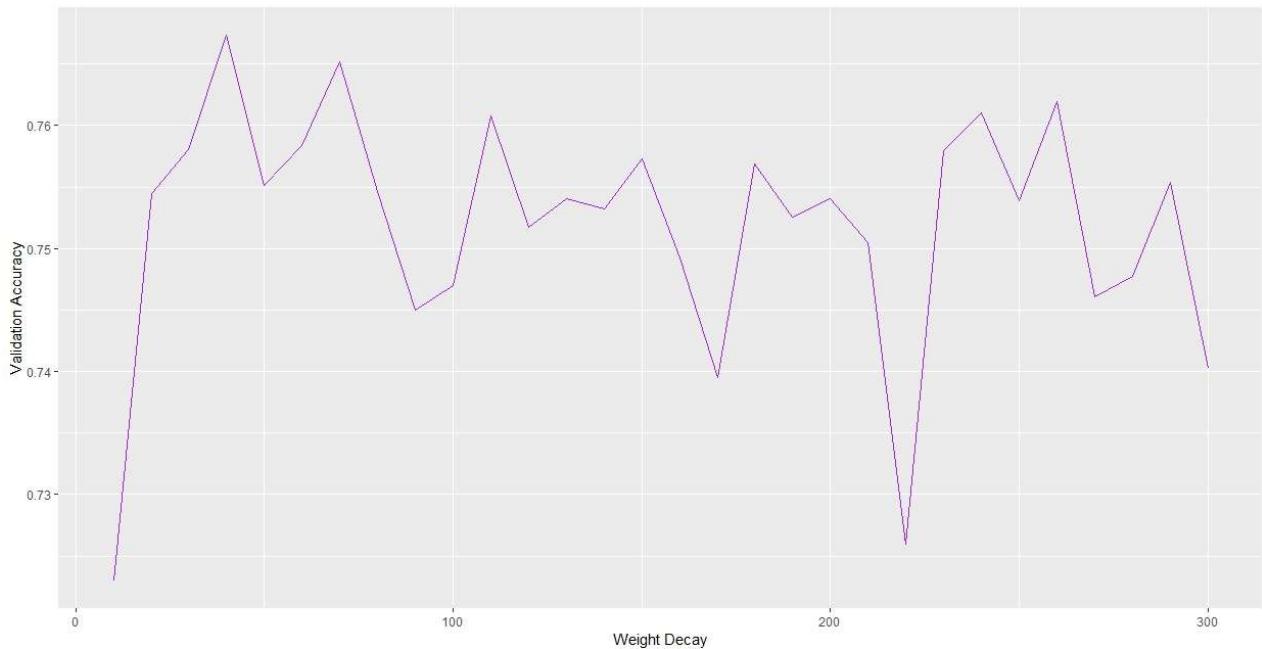


נספח 8 – מספר משקלות נלמדות בקונפיגורציות השונות

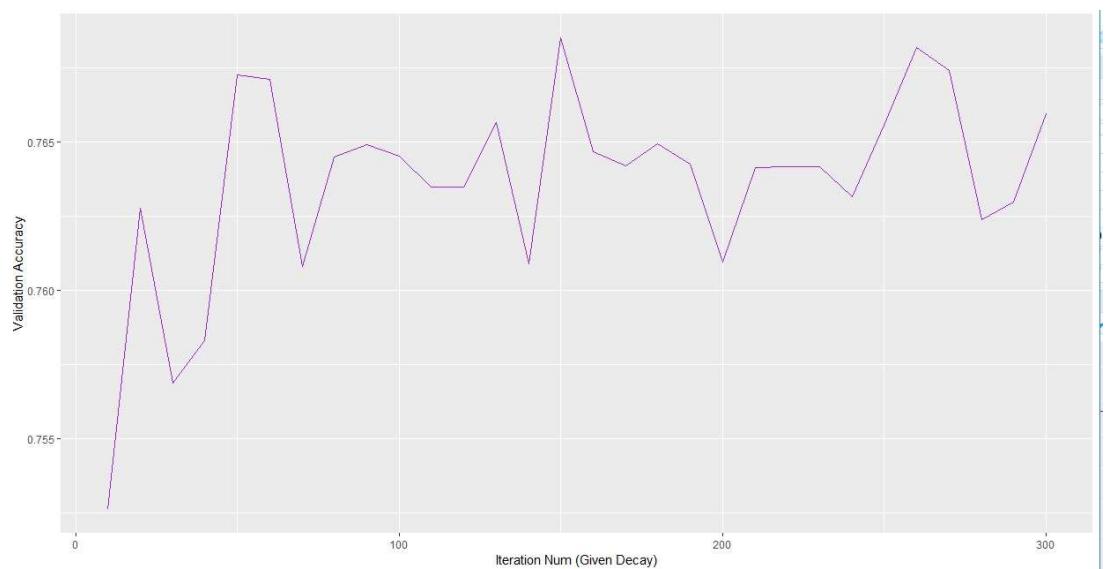


$$2 \times 22 \times 2 + 1 \times 2 \text{ biases} + 1 \times 2 \text{ biases} = 52 \text{ weight}$$

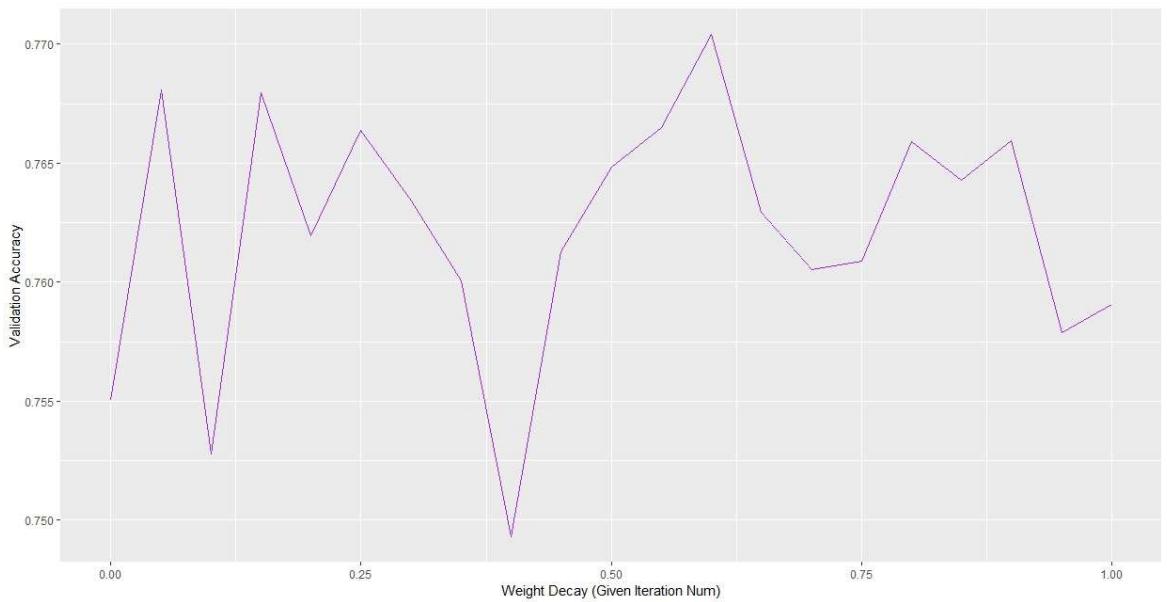
נספח 9 – ערך אופטימאלי קונפיגורציה מספר איטרציות



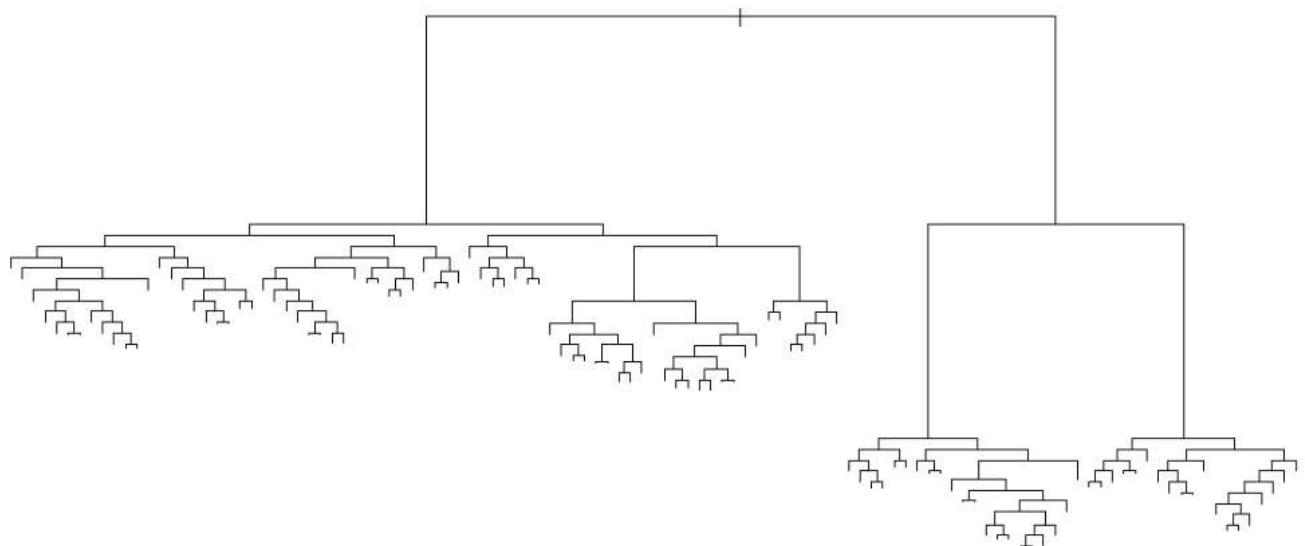
נוגה 10 – מספר איטרציות אופטימאלי, לערך decay נתון 0.85



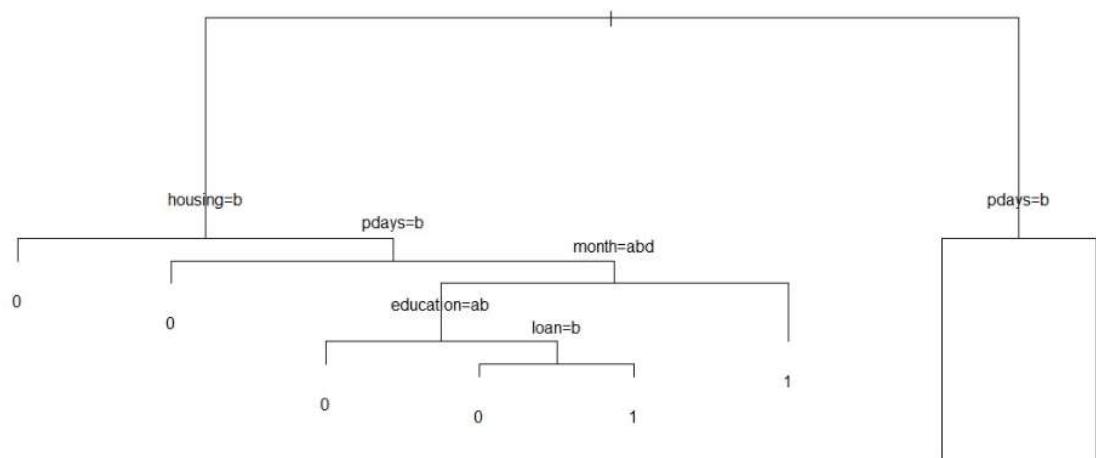
נוגה 11 – ערך decay אופטימאלי למספר איטרציות נתון 40



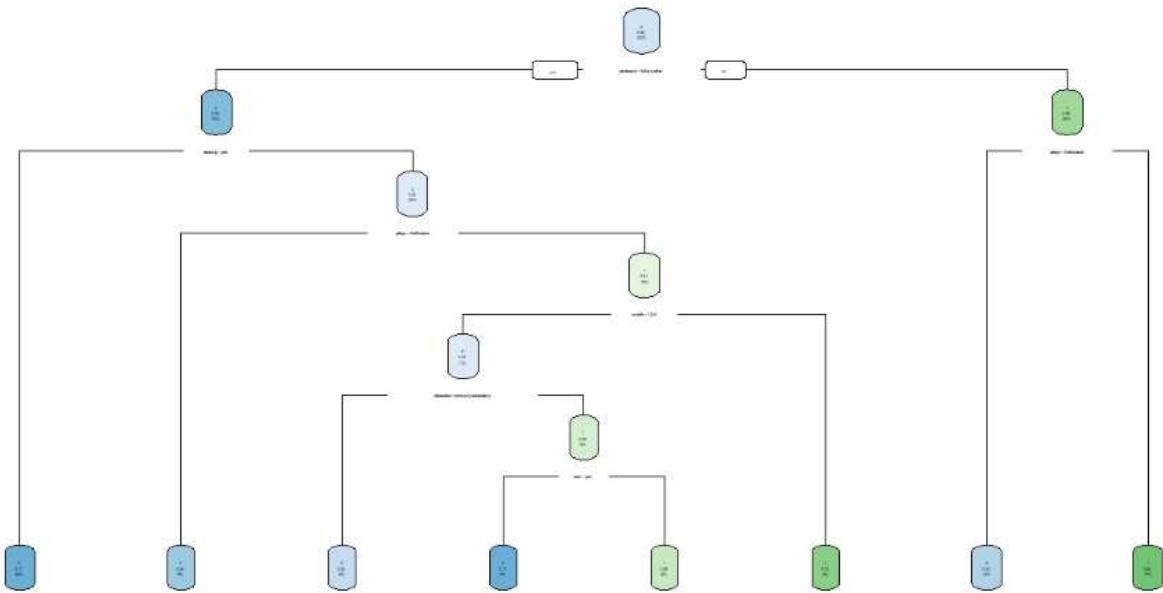
נספח 12 – עץ החלטה לפני קטימה



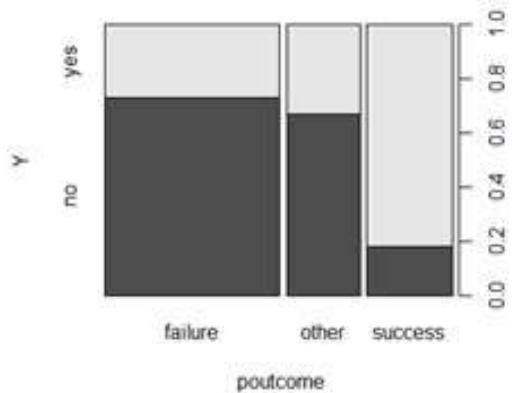
נספח 13 – עץ החלטה לאחר קטימה לפי ערך cp אופטימאלי



נספח 14 – פלט העץ לאחר קטימה בעזרת שימוש בrpart.

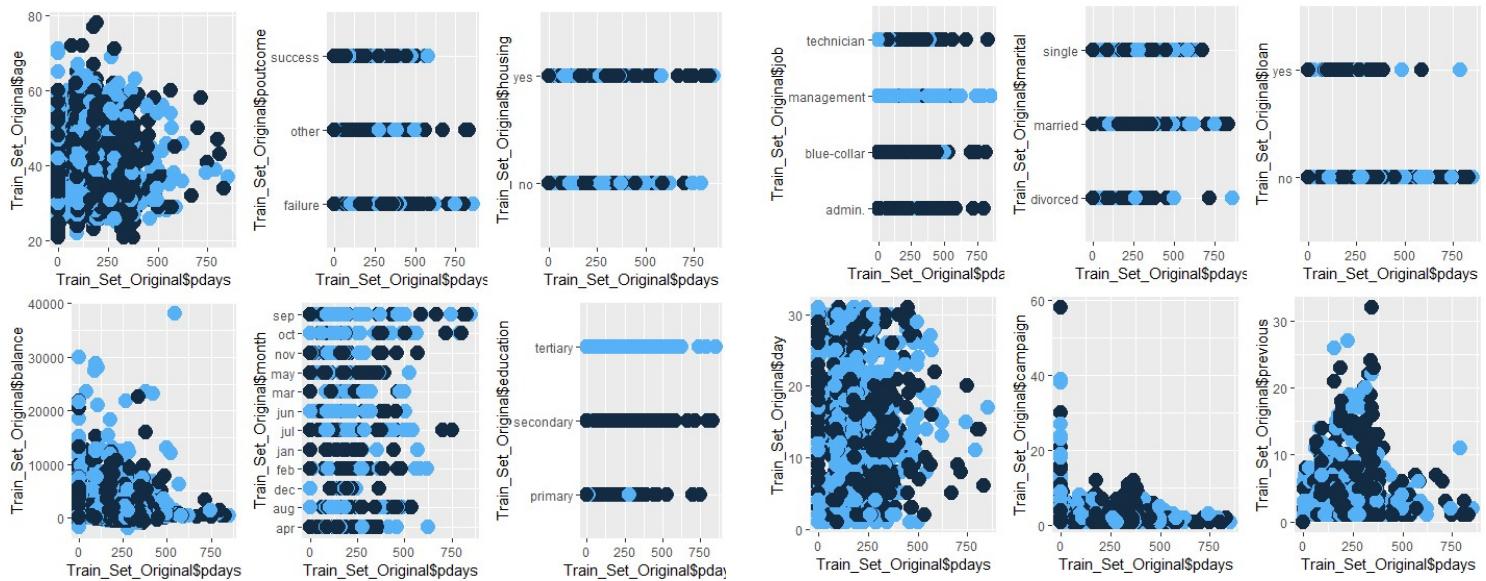


נספח 15 – קשר בין מאפיין `poutcome` למשתנה המטרה `y` שנמצא בחלק א'

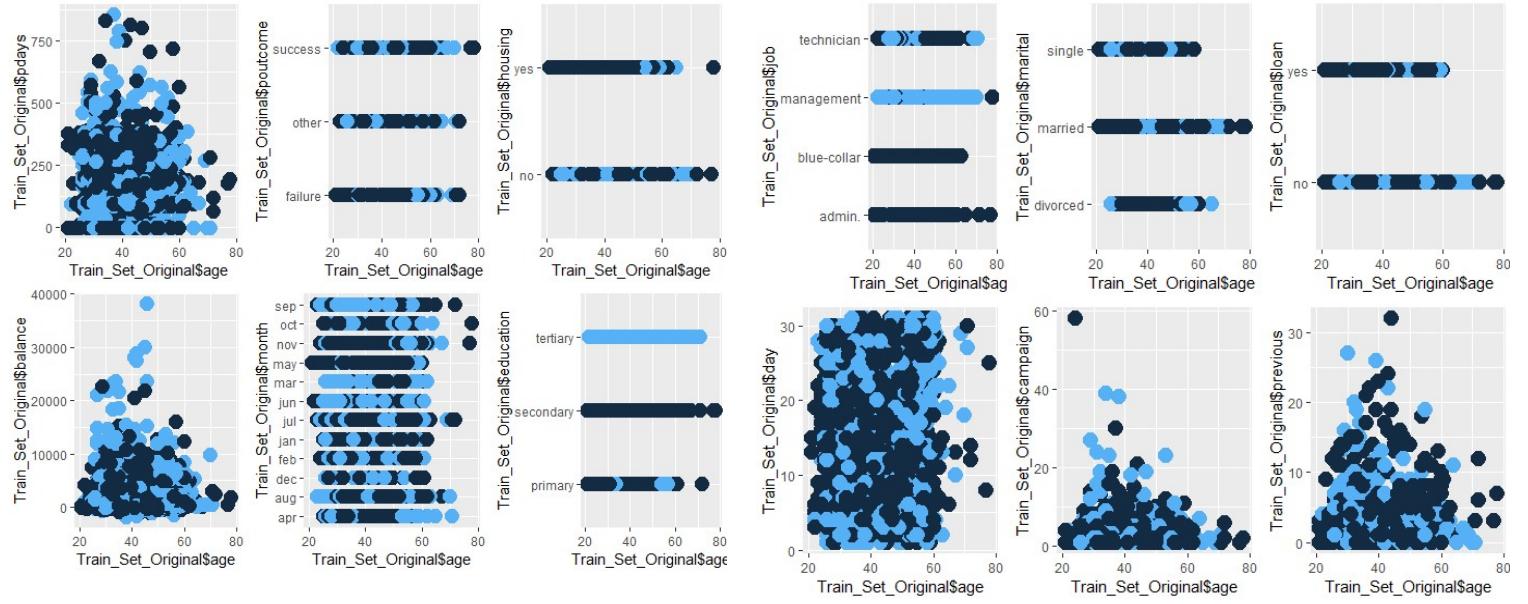


נספח 16 – ניתוח מאפיינים שונים בהתאם למסוגים

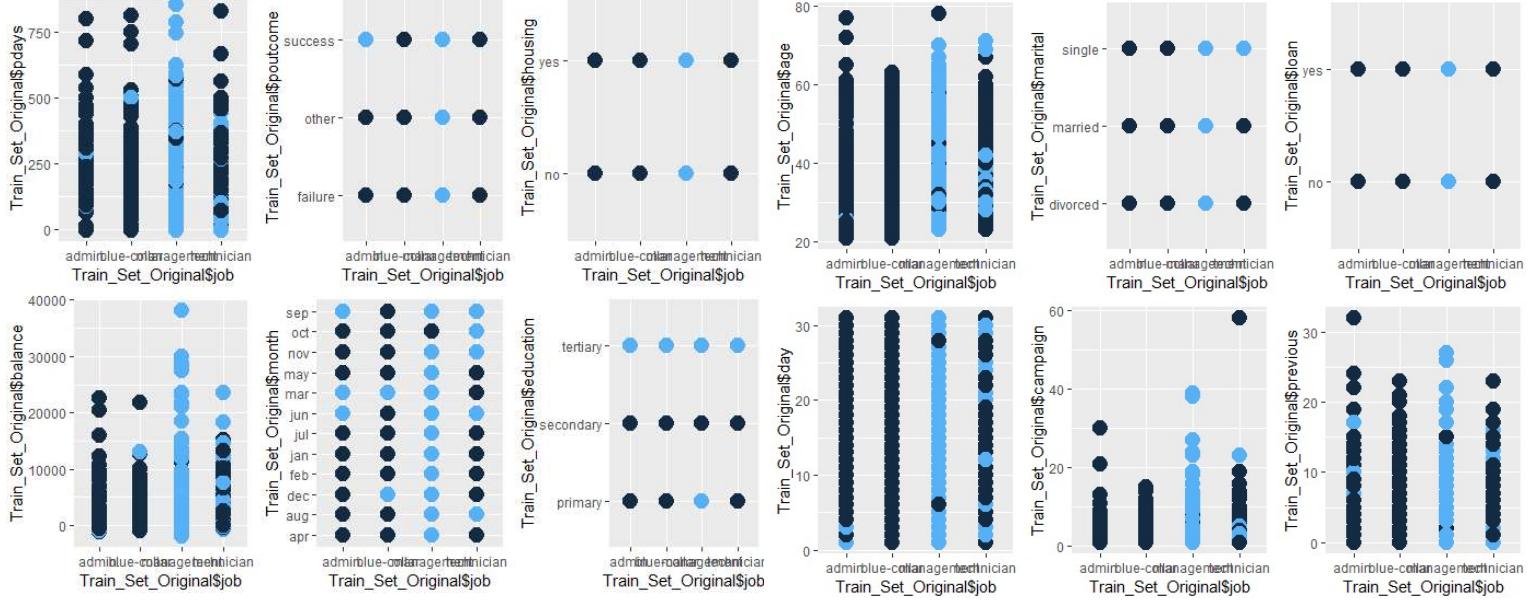
:Pdays



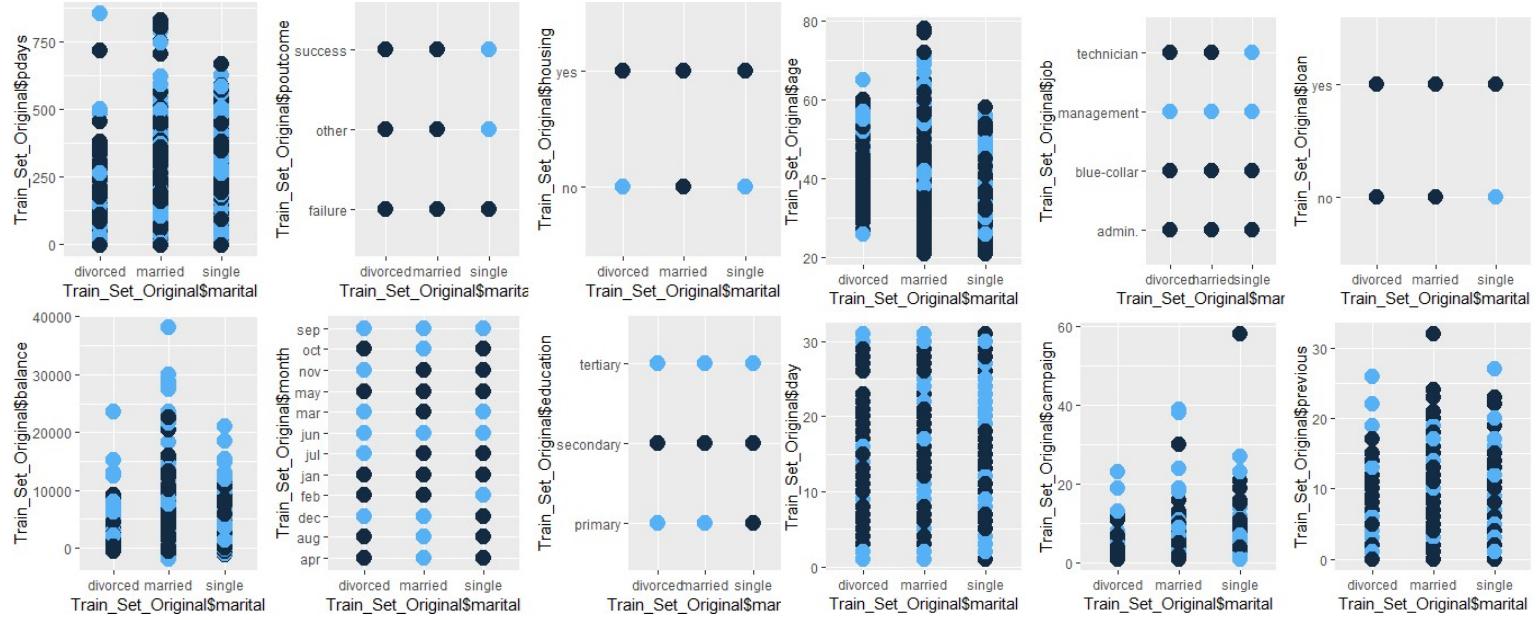
:Age



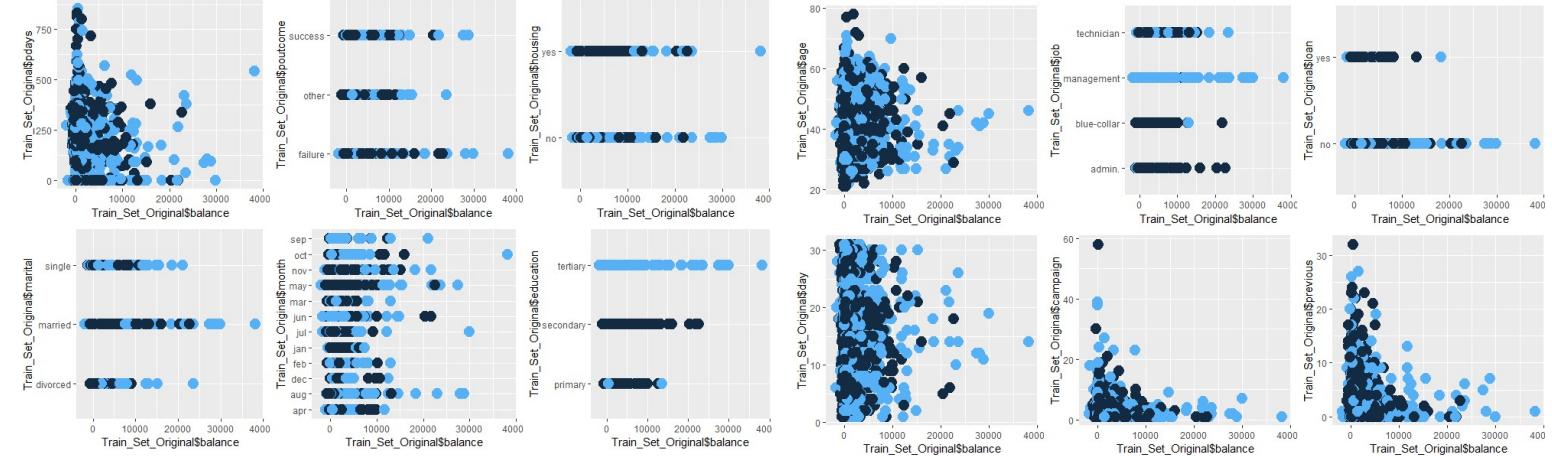
:Job:



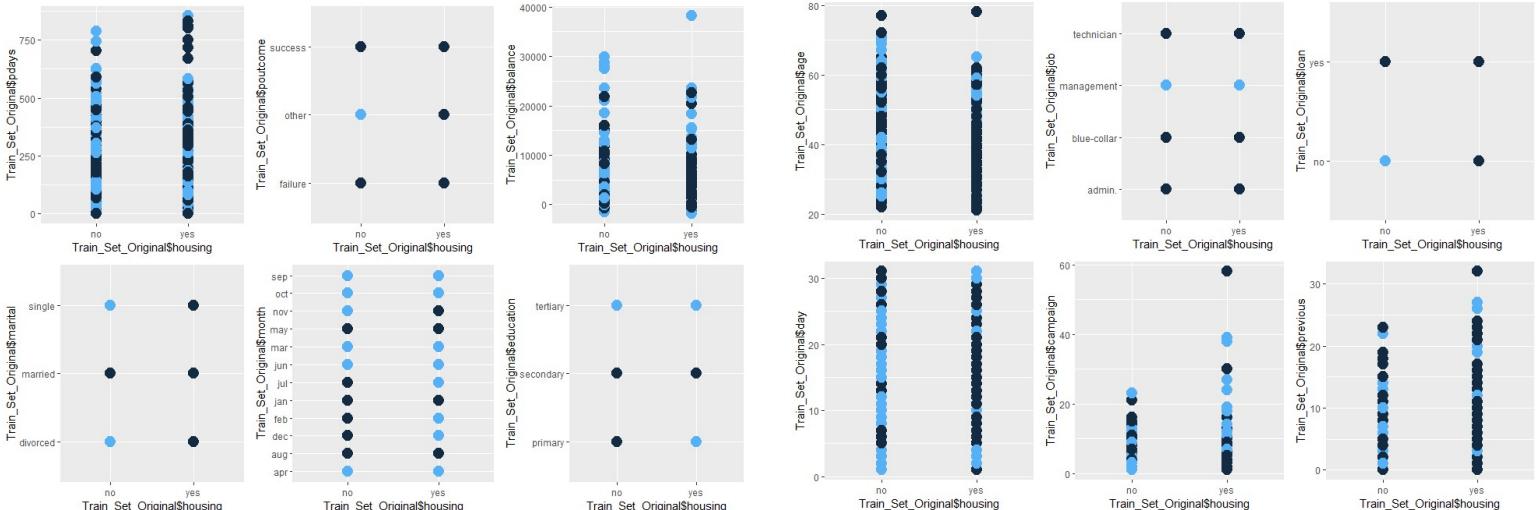
:Marital



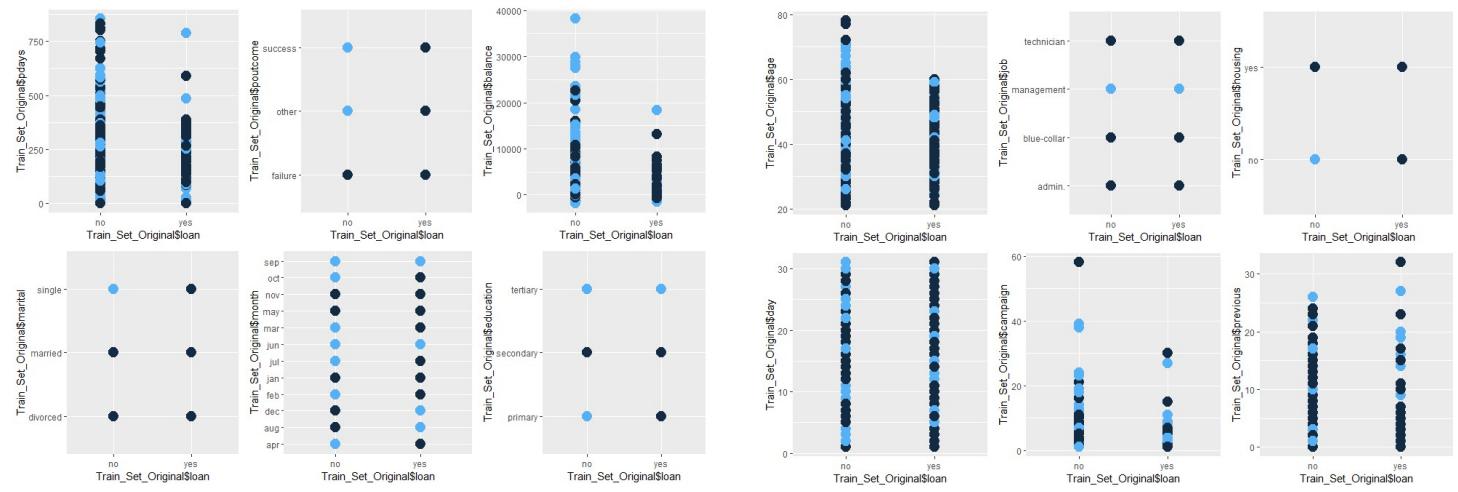
:Balance



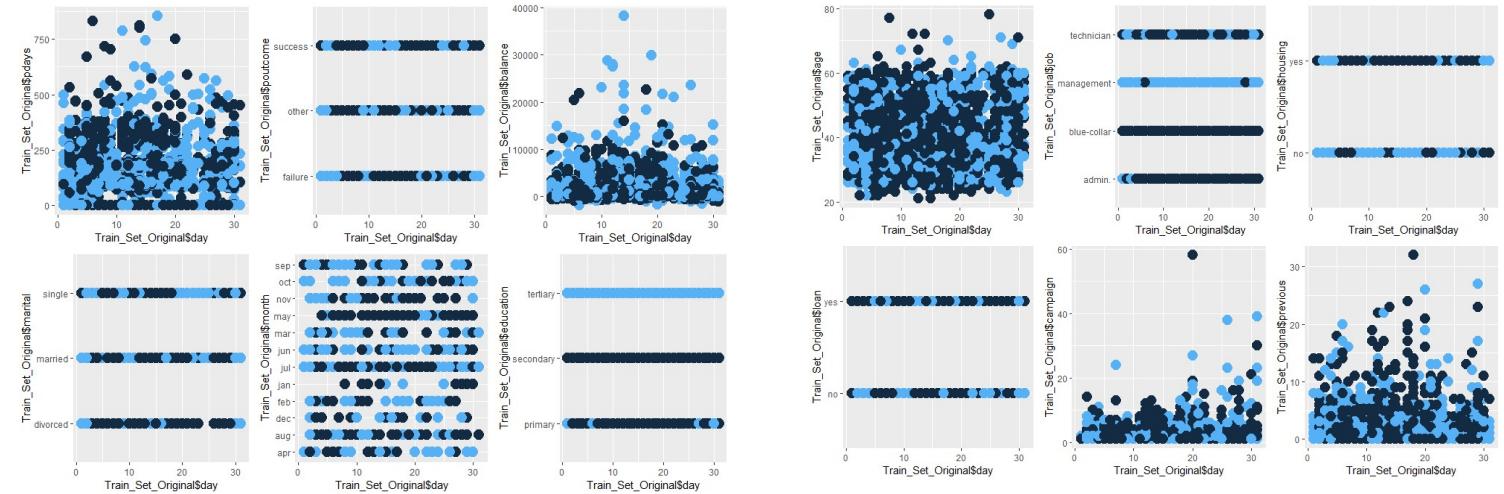
:Housing



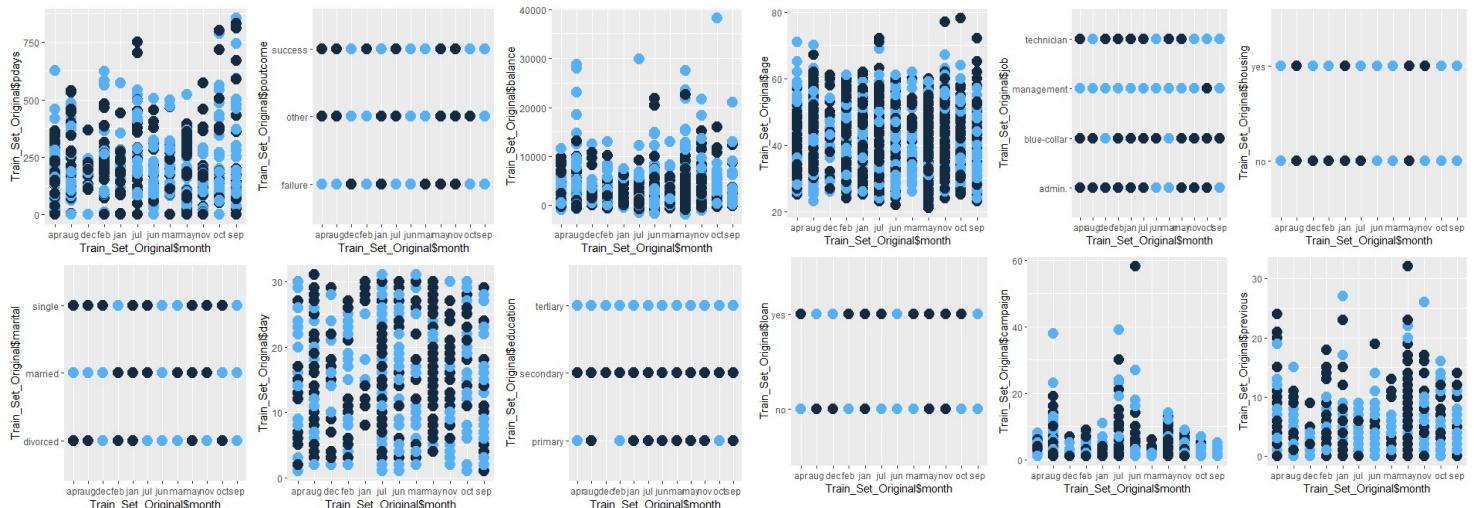
:Loan



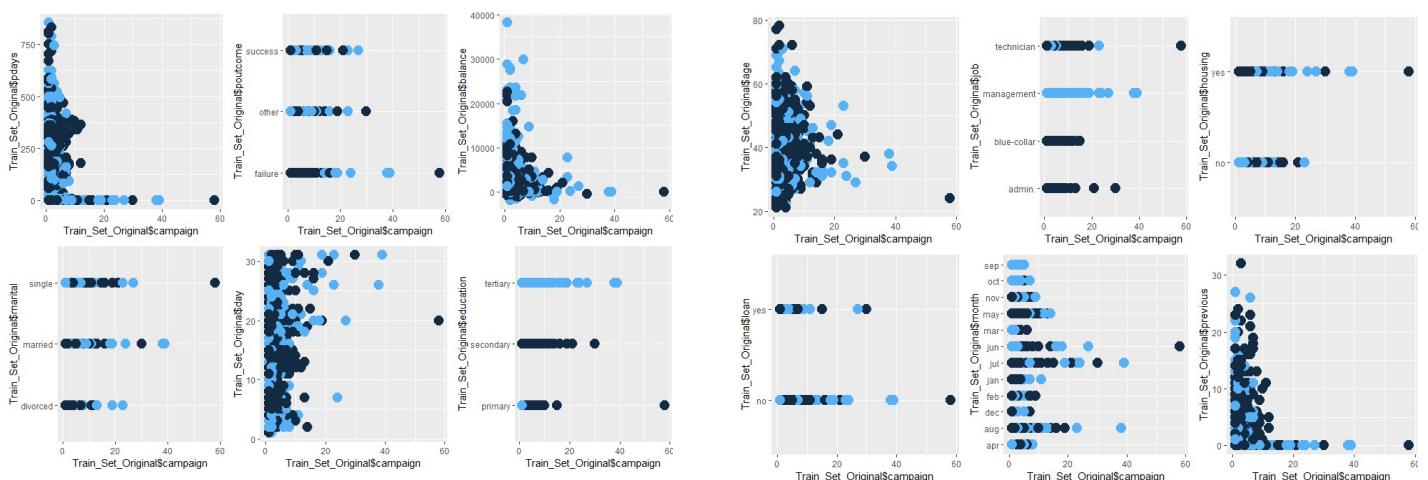
:Day



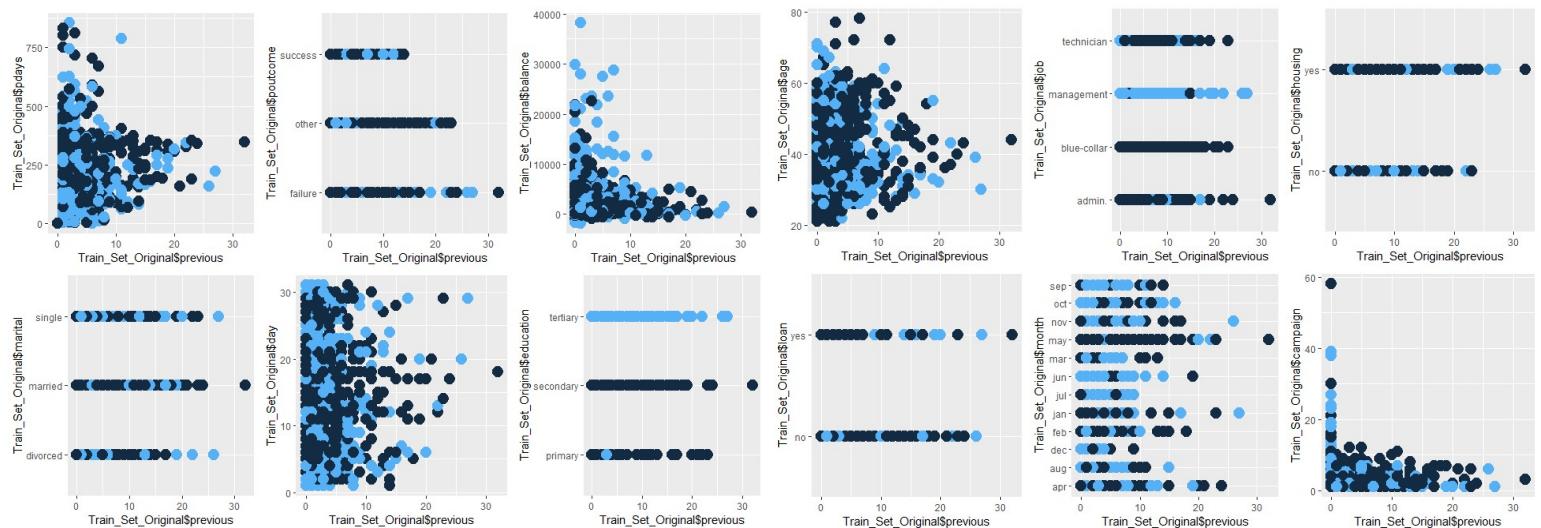
:Month



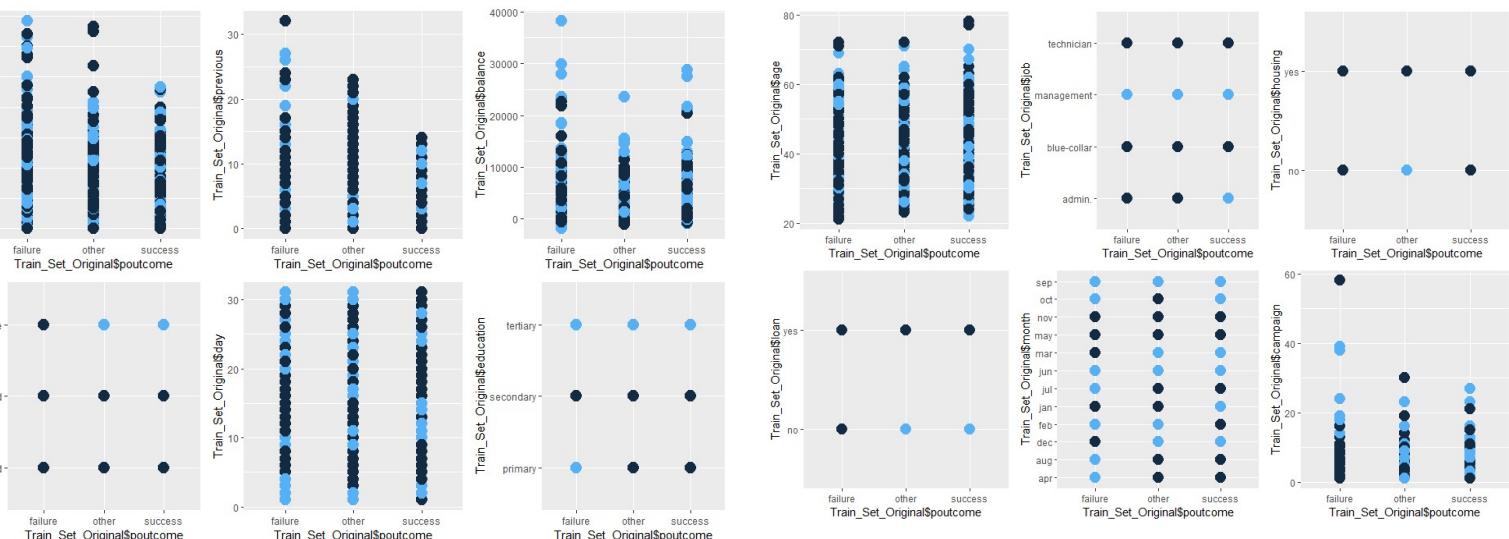
:Campaign



:Previous



:Poutcome



```

# SVM
#-----
## classification mode
# default with factor response:
model_svm <- svm(as.factor(Train_Set$y) ~ ., data = Train_Set)

print(model_svm)
summary(model_svm)

# train perciosion
pred_train_svm <- predict(model_svm, Train_Set[,1:22])
pred_train_svm <- c(pred_train_svm)
pred_train_svm <- pred_train_svm-1
svm_train_perciosin <- sum(pred_train_svm==Train_Set$y)/nrow(Train_Set)

# test perciosion
pred_test_svm <- predict(model_svm, Test_Set[,1:22])
pred_test_svm <- c(pred_test_svm)
pred_test_svm <- pred_test_svm-1
svm_test_perciosin <- sum(pred_test_svm==Test_Set$y)/nrow(Test_Set)

```

נוף 18 – קוד העבודה

```

#-----

filePath=choose.files()

Bank_Data_Original<-read.csv(filePath,header=TRUE)

# complete unknown data

Bank_Data_Original[Bank_Data_Original=="unknown"] <- NA

Bank_Data_Original <- mice(data=Bank_Data_Original, m=5, method="pmm", maxit=50, seed=50)

Bank_Data_Original <- complete(Bank_Data_Original)

Bank_Data <- Bank_Data_Original

# month to quarters

Bank_Data$month <- as.character(Bank_Data$month)

Bank_Data$month[Bank_Data$month=="jan" | Bank_Data$month=="feb" | Bank_Data$month=="mar"] <- 1
Bank_Data$month[Bank_Data$month=="apr" | Bank_Data$month=="may" | Bank_Data$month=="jun"] <- 2
Bank_Data$month[Bank_Data$month=="jul" | Bank_Data$month=="aug" | Bank_Data$month=="sep" ] <- 3
Bank_Data$month[Bank_Data$month=="oct" | Bank_Data$month=="nov" | Bank_Data$month=="dec" ] <- 4

Bank_Data$month <- as.factor(Bank_Data$month)
```

```
# pdays to categories

Bank_Data$pdays <- findInterval(Bank_Data$pdays,c(-1,0,181,366,1000))

Bank_Data$pdays[Bank_Data$pdays==4] <- "OverOneYear"

Bank_Data$pdays[Bank_Data$pdays==3 ] <- "OneYear"

Bank_Data$pdays[Bank_Data$pdays==2] <- "HalfYear"

Bank_Data$pdays[Bank_Data$pdays==1] <- "NoContact"
```

```
# Dummy variables
```

```
#pdays

Bank_Data$pdays_D1 <- ifelse(Bank_Data$pdays=="HalfYear",1,-1)

Bank_Data$pdays_D2 <- ifelse(Bank_Data$pdays=="OneYear",1,-1)

Bank_Data$pdays_D3 <- ifelse(Bank_Data$pdays=="OverOneYear",1,-1)
```

```
Bank_Data$pdays_D1 <- as.factor(Bank_Data$pdays_D1)

Bank_Data$pdays_D2 <- as.factor(Bank_Data$pdays_D2)

Bank_Data$pdays_D3 <- as.factor(Bank_Data$pdays_D3)
```

```
# job
```

```
Bank_Data$job_D1 <- ifelse(Bank_Data$job=="blue-collar",1,-1)

Bank_Data$job_D2 <- ifelse(Bank_Data$job=="management",1,-1)

Bank_Data$job_D3 <- ifelse(Bank_Data$job=="technician",1,-1)
```

```
Bank_Data$job_D1 <- as.factor(Bank_Data$job_D1)

Bank_Data$job_D2 <- as.factor(Bank_Data$job_D2)

Bank_Data$job_D3 <- as.factor(Bank_Data$job_D3)
```

```
#marital
```

```
Bank_Data$marital_D1 <- ifelse(Bank_Data$marital=="married",1,-1)

Bank_Data$marital_D2 <- ifelse(Bank_Data$marital=="divorced",1,-1)
```

```
Bank_Data$marital_D1 <- as.factor(Bank_Data$marital_D1)
Bank_Data$marital_D2 <- as.factor(Bank_Data$marital_D2)

#education
Bank_Data$education_D1 <- ifelse(Bank_Data$education=="secondary",1,-1)
Bank_Data$education_D2 <- ifelse(Bank_Data$education=="tertiary",1,-1)

Bank_Data$education_D1 <- as.factor(Bank_Data$education_D1)
Bank_Data$education_D2 <- as.factor(Bank_Data$education_D2)

#month
Bank_Data$Quarter_D1 <- ifelse(Bank_Data$month=="2",1,-1)
Bank_Data$Quarter_D2 <- ifelse(Bank_Data$month=="3",1,-1)
Bank_Data$Quarter_D3 <- ifelse(Bank_Data$month=="4",1,-1)

Bank_Data$Quarter_D1 <- as.factor(Bank_Data$Quarter_D1)
Bank_Data$Quarter_D2 <- as.factor(Bank_Data$Quarter_D2)
Bank_Data$Quarter_D3 <- as.factor(Bank_Data$Quarter_D3)

#poutcome
Bank_Data$poutcome_D1 <- ifelse(Bank_Data$poutcome=="failure",1,-1)
Bank_Data$poutcome_D2 <- ifelse(Bank_Data$poutcome=="other",1,-1)

Bank_Data$poutcome_D1 <- as.factor(Bank_Data$poutcome_D1)
Bank_Data$poutcome_D2 <- as.factor(Bank_Data$poutcome_D2)

#housing
Bank_Data$housing_D1 <- ifelse(Bank_Data$housing=="yes",1,-1)

Bank_Data$housing_D1 <- as.factor(Bank_Data$housing_D1)

#loan
Bank_Data$loan_D1 <- ifelse(Bank_Data$loan=="yes",1,-1)
```

```

Bank_Data$loan_D1 <- as.factor(Bank_Data$loan_D1)

# columns deletion
deletion <- c(2,3,4,6,7,9,11,13)
Bank_Data <- Bank_Data[,-deletion]
y <- Bank_Data$y
Bank_Data <- Bank_Data[,-6]
Bank_Data <- as.data.frame(cbind(Bank_Data,y))

# scaling
Bank_Data$age <- scale(Bank_Data$age)
Bank_Data$balance <- scale(Bank_Data$balance)
Bank_Data$day <- scale(Bank_Data$day)
Bank_Data$campaign <- scale(Bank_Data$campaign)
Bank_Data$previous <- scale(Bank_Data$previous)

#-----
# divide to train and test sets

set.seed(123)
Rows_Numbers_for_Test_Set <- sample(1:nrow(Bank_Data),0.2*nrow(Bank_Data),replace = FALSE)

Train_Set <- Bank_Data[-Rows_Numbers_for_Test_Set,]
Test_Set <- Bank_Data[Rows_Numbers_for_Test_Set,]

# K-fold configuration
n.folds <- 5

```

```

samples <- sample(1:n.folds, nrow(Train_Set), replace=T)

Train_and_Val <- cbind(Train_Set, samples)

k_fold_sets <- list()

for(fold in 1:n.folds){

  k_fold_sets[[fold]] <- Train_and_Val[Train_and_Val$samples==fold,1:ncol(Train_and_Val)-1]

}

k_folds_Train_sets <- list()

k_folds_Valid_sets <- list()

k_folds_Train_sets[[1]] <- rbind(k_fold_sets[[2]],k_fold_sets[[3]],k_fold_sets[[4]],k_fold_sets[[5]])

k_folds_Train_sets[[2]] <- rbind(k_fold_sets[[1]],k_fold_sets[[3]],k_fold_sets[[4]],k_fold_sets[[5]])

k_folds_Train_sets[[3]] <- rbind(k_fold_sets[[1]],k_fold_sets[[2]],k_fold_sets[[4]],k_fold_sets[[5]])

k_folds_Train_sets[[4]] <- rbind(k_fold_sets[[1]],k_fold_sets[[2]],k_fold_sets[[3]],k_fold_sets[[5]])

k_folds_Train_sets[[5]] <- rbind(k_fold_sets[[1]],k_fold_sets[[2]],k_fold_sets[[3]],k_fold_sets[[4]])


k_folds_Valid_sets[[1]] <- k_fold_sets[[1]]

k_folds_Valid_sets[[2]] <- k_fold_sets[[2]]

k_folds_Valid_sets[[3]] <- k_fold_sets[[3]]

k_folds_Valid_sets[[4]] <- k_fold_sets[[4]]

k_folds_Valid_sets[[5]] <- k_fold_sets[[5]]


#-----



#-----



# neural networks



#-----



# 1



set.seed(123)

nn <- nnet(x=Train_Set[,1:22], y=class.ind(Train_Set[,23]), size=1, linout=FALSE, softmax=T) # train

```

```

preds_nn_train <- factor(predict(nn, newdata=Train_Set[,1:22], type='class')) # prediction for train set
preds_nn_test <- factor(predict(nn, newdata=Test_Set[,1:22], type='class')) # prediction for test set

nn_default_train_percision <- sum(preds_nn_train==Train_Set$y)/nrow(Train_Set)
nn_default_test_percision <- sum(preds_nn_test==Test_Set$y)/nrow(Test_Set)

plotnet(nn)
summary(nn)
olden(nn)
# olden(nn, out_var=Train_Set[,23])

# 2

#nnum <- seq(1,101,5)
nnum <- c(1,2,3,4,5,6,7,8,9,10)
acc <- matrix(0,length(nnum),1)
i <- 1
for(neurons in nnum){
  temp_total_precision <- 0
  for (fold in 1:n.folds){
    nn <- nnet(x=k_folds_Train_sets[[fold]][,1:22], y=class.ind(k_folds_Train_sets[[fold]]$y), size=neurons,
               linout=FALSE, softmax=T, MaxNWts=100000)

    preds_nn_val <- factor(predict(nn, newdata=k_folds_Valid_sets[[fold]][,1:22], type='class'))
    temp_total_precision <-
    temp_total_precision+(sum(preds_nn_val==k_folds_Valid_sets[[fold]]$y)/nrow(k_folds_Valid_sets[[fold]]))
  }
  acc[i] <- temp_total_precision/n.folds
  i <- i + 1
}

ggplot(data.frame(x=nnum, y=acc)) + geom_line(aes(x,y), color='purple') +

```

```

xlab("Number of Neurons") + ylab("Validation Accuracy")

print(acc)

neurons <- nnum[which.max(acc)] # find the size with maximum accuracy

valid_percision_after_neurons_num <- max(acc)

set.seed(123)

nn <- nnet(x=Train_Set[,1:22], y=class.ind(Train_Set$y), size=neurons, linout=FALSE, softmax=T) # train

preds_nn_train <- factor(predict(nn, newdata=Train_Set[,1:22], type='class')) # prediction for train set

preds_nn_test <- factor(predict(nn, newdata=Test_Set[,1:22], type='class')) # prediction for test set

nn_after_neuronsNum_train_percision <- sum(preds_nn_train==Train_Set$y)/nrow(Train_Set)

nn_after_neuronsNum_test_percision <- sum(preds_nn_test==Test_Set$y)/nrow(Test_Set)

# 3

# configuration 1- weight decay

nnum <- seq(0,1,0.05)

decay_val <- matrix(0,length(nnum),1)

i <- 1

for(n_decay in nnum){

  temp_total_precision <- 0

  for (fold in 1:n.folds){

    nn <- nnet(x=k_folds_Train_sets[[fold]][,1:22], y=class.ind(k_folds_Train_sets[[fold]]$y), size=neurons,
    linout=FALSE, softmax=T, MaxNWts=100000, decay = n_decay )

    preds_nn_val <- factor(predict(nn, newdata=k_folds_Valid_sets[[fold]][,1:22], type='class'))

    temp_total_precision <-
    temp_total_precision+(sum(preds_nn_val==k_folds_Valid_sets[[fold]]$y)/nrow(k_folds_Valid_sets[[fold]]))

  }

  decay_val[i] <- temp_total_precision/n.folds
}

```

```

i      <- i + 1
}

ggplot(data.frame(x=nnum, y=decay_val)) + geom_line(aes(x,y), color='purple') +
xlab("Weight Decay") + ylab("Validation Accuracy")

weight_decay <- nnum[which.max(decay_val)]

nn_Configure1_valid_percision <- max(decay_val)

nn <- nnet(x=k_folds_Train_sets[[1]][,1:22], y=class.ind(k_folds_Train_sets[[1]]$y), size=neurons, linout=FALSE,
softmax=T, MaxNWts=100000, decay = weight_decay )

print(nn)

plotnet(nn)

# curr_fold <- 2

# nn <- nnet(x=k_folds_Train_sets[[curr_fold]][,1:22], y=class.ind(k_folds_Train_sets[[curr_fold]]$y), size=neurons,
linout=FALSE, softmax=T, MaxNWts=100000, decay = weight_decay )

# preds_nn_train <- factor(predict(nn, newdata=k_folds_Train_sets[[curr_fold]][,1:22], type='class')) # prediction for
train set

# preds_nn_valid <- factor(predict(nn, newdata=k_folds_Valid_sets[[curr_fold]][,1:22], type='class'))

# preds_nn_test <- factor(predict(nn, newdata=Test_Set[,1:22], type='class')) # prediction for test set

# nn_Configure1_train_percision <-
sum(preds_nn_train==k_folds_Train_sets[[curr_fold]]$y)/nrow(k_folds_Train_sets[[curr_fold]])

# nn_Configure1_valid_percision <-
sum(preds_nn_valid==k_folds_Valid_sets[[curr_fold]]$y)/nrow(k_folds_Valid_sets[[curr_fold]])

# nn_Configure1_test_percision <- sum(preds_nn_test==Test_Set$y)/nrow(Test_Set)

set.seed(123)

nn <- nnet(x=Train_Set[,1:22], y=class.ind(Train_Set[,23]), size=neurons, linout=FALSE, softmax=T, decay =
weight_decay ) # train

preds_nn_train <- factor(predict(nn, newdata=Train_Set[,1:22], type='class')) # prediction for train set

preds_nn_test <- factor(predict(nn, newdata=Test_Set[,1:22], type='class')) # prediction for test set

nn_Configure1_train_percision <- sum(preds_nn_train==Train_Set$y)/nrow(Train_Set)

nn_Configure1_test_percision <- sum(preds_nn_test==Test_Set$y)/nrow(Test_Set)

# configuration 2- iteration number

```

```

nnum <- seq(10,300,10)

iteration_val <- matrix(0,length(nnum),1)

i <- 1

for(n_iteration in nnum){

  temp_total_precision <- 0

  for (fold in 1:n.folds){

    nn <- nnet(x=k_folds_Train_sets[[fold]][,1:22], y=class.ind(k_folds_Train_sets[[fold]]$y), size=neurons,
linout=FALSE, softmax=T, MaxNWts=100000, maxit = n_iteration )

    preds_nn_val <- factor(predict(nn, newdata=k_folds_Valid_sets[[fold]][,1:22], type='class'))

    temp_total_precision <-
temp_total_precision+(sum(preds_nn_val==k_folds_Valid_sets[[fold]]$y)/nrow(k_folds_Valid_sets[[fold]]))

  }

  iteration_val[i] <- temp_total_precision/n.folds

  i <- i + 1

}

ggplot(data.frame(x=nnum, y=iteration_val)) + geom_line(aes(x,y), color='purple') +
xlab("Weight Decay") + ylab("Validation Accuracy")

iteration_number <- nnum[which.max(iteration_val)]

nn_Configure2_valid_percision <- max(iteration_val)

nn <- nnet(x=k_folds_Train_sets[[1]][,1:22], y=class.ind(k_folds_Train_sets[[1]]$y), size=neurons, linout=FALSE,
softmax=T, MaxNWts=100000, maxit = iteration_number )

print(nn)

plotnet(nn)

set.seed(123)

nn <- nnet(x=Train_Set[,1:22], y=class.ind(Train_Set[,23]), size=neurons, linout=FALSE, softmax=T, maxit =
iteration_number ) # train

preds_nn_train <- factor(predict(nn, newdata=Train_Set[,1:22], type='class')) # prediction for train set

preds_nn_test <- factor(predict(nn, newdata=Test_Set[,1:22], type='class')) # prediction for test set

nn_Configure2_train_percision <- sum(preds_nn_train==Train_Set$y)/nrow(Train_Set)

nn_Configure2_test_percision <- sum(preds_nn_test==Test_Set$y)/nrow(Test_Set)

```

```

# configuration 3- combind between configuration 1 and 2

# finding the iteration number given weight decay

nnum <- seq(10,300,10)

iteration_val <- matrix(0,length(nnum),1)

i <- 1

for(n_iteration in nnum){

  temp_total_precision <- 0

  for (fold in 1:n.folds){

    nn <- nnet(x=k_folds_Train_sets[[fold]][,1:22], y=class.ind(k_folds_Train_sets[[fold]]$y), size=neurons,
linout=FALSE, softmax=T, MaxNWts=100000, maxit = n_iteration, decay = weight_decay )

    preds_nn_val <- factor(predict(nn, newdata=k_folds_Valid_sets[[fold]][,1:22], type='class'))

    temp_total_precision <-
temp_total_precision+(sum(preds_nn_val==k_folds_Valid_sets[[fold]]$y)/nrow(k_folds_Valid_sets[[fold]]))

  }

  iteration_val[i] <- temp_total_precision/n.folds

  i <- i + 1

}

ggplot(data.frame(x=nnum, y=iteration_val)) + geom_line(aes(x,y), color='purple') +

xlab("Iteration Num (Given Decay)") + ylab("Validation Accuracy")

iteration_number <- nnum[which.max(iteration_val)]

set.seed(123)

nn <- nnet(x=Train_Set[,1:22], y=class.ind(Train_Set[,23]), size=neurons, linout=FALSE, softmax=T, maxit =
iteration_number, decay = weight_decay ) # train

preds_nn_train <- factor(predict(nn, newdata=Train_Set[,1:22], type='class')) # prediction for train set

preds_nn_test <- factor(predict(nn, newdata=Test_Set[,1:22], type='class')) # prediction for test set

nn_Configure3A_train_percision <- sum(preds_nn_train==Train_Set$y)/nrow(Train_Set)

nn_Configure3A_test_percision <- sum(preds_nn_test==Test_Set$y)/nrow(Test_Set)

```

```

# finding weight decay given iteration number

nnum <- seq(0,1,0.05)

iteration_number <- 40

decay_val <- matrix(0,length(nnum),1)

i <- 1

for(n_decay in nnum){

  temp_total_precision <- 0

  for (fold in 1:n.folds){

    nn <- nnet(x=k_folds_Train_sets[[fold]][,1:22], y=class.ind(k_folds_Train_sets[[fold]]$y), size=neurons,
    linout=FALSE, softmax=T, MaxNWts=100000,maxit = iteration_number, decay = n_decay )

    preds_nn_val <- factor(predict(nn, newdata=k_folds_Valid_sets[[fold]][,1:22], type='class'))

    temp_total_precision <-
    temp_total_precision+(sum(preds_nn_val==k_folds_Valid_sets[[fold]]$y)/nrow(k_folds_Valid_sets[[fold]]))

  }

  decay_val[i] <- temp_total_precision/n.folds

  i <- i + 1

}

ggplot(data.frame(x=nnum, y=decay_val)) + geom_line(aes(x,y), color='purple') +
  xlab("Weight Decay (Given Iteration Num)") + ylab("Validation Accuracy")

weight_decay <- nnum[which.max(decay_val)]

set.seed(123)

nn <- nnet(x=Train_Set[,1:22], y=class.ind(Train_Set[,23]), size=neurons, linout=FALSE, softmax=T, maxit =
iteration_number, decay = weight_decay ) # train

preds_nn_train <- factor(predict(nn, newdata=Train_Set[,1:22], type='class')) # prediction for train set

preds_nn_test <- factor(predict(nn, newdata=Test_Set[,1:22], type='class')) # prediction for test set

nn_Configure3B_train_percision <- sum(preds_nn_train==Train_Set$y)/nrow(Train_Set)

nn_Configure3B_test_percision <- sum(preds_nn_test==Test_Set$y)/nrow(Test_Set)

## The Chosen Model

```

```
set.seed(123)

nn_chosen <- nnet(x=Train_Set[,1:22], y=class.ind(Train_Set[,23]), size=2, linout=FALSE, softmax=T, maxit = 40,
decay = 0.6 ) # train

preds_nn_train <- factor(predict(nn_chosen, newdata=Train_Set[,1:22], type='class')) # prediction for train set

preds_nn_test <- factor(predict(nn_chosen, newdata=Test_Set[,1:22], type='class')) # prediction for test set

nn_Configure3B_train_percision <- sum(preds_nn_train==Train_Set$y)/nrow(Train_Set)

nn_Configure3B_test_percision <- sum(preds_nn_test==Test_Set$y)/nrow(Test_Set)
```

#confusion matrix- Chosen Model

```
table(prediction = preds_nn_test, true_values = Test_Set$y)
```

#-----

Decision Trees

#-----

```
Bank_Data_Trees <- Bank_Data_Original
```

making the catagorial variables

month to quarters

```
Bank_Data_Trees$month <- as.character(Bank_Data_Trees$month)

Bank_Data_Trees$month[Bank_Data_Trees$month=="jan" | Bank_Data_Trees$month=="feb" |
Bank_Data_Trees$month=="mar"] <- 1

Bank_Data_Trees$month[Bank_Data_Trees$month=="apr" | Bank_Data_Trees$month=="may" |
Bank_Data_Trees$month=="jun"] <- 2

Bank_Data_Trees$month[Bank_Data_Trees$month=="jul" | Bank_Data_Trees$month=="aug" |
Bank_Data_Trees$month=="sep" ] <- 3

Bank_Data_Trees$month[Bank_Data_Trees$month=="oct" | Bank_Data_Trees$month=="nov" |
Bank_Data_Trees$month=="dec" ] <- 4
```

```
Bank_Data_Trees$month <- as.factor(Bank_Data_Trees$month)
```

pdays to categories

```
Bank_Data_Trees$pdays <- findInterval(Bank_Data_Trees$pdays,c(-1,0,181,366,1000))
```

```

Bank_Data_Trees$pdays[Bank_Data_Trees$pdays==4] <- "OverOneYear"
Bank_Data_Trees$pdays[Bank_Data_Trees$pdays==3 ] <- "OneYear"
Bank_Data_Trees$pdays[Bank_Data_Trees$pdays==2] <- "HalfYear"
Bank_Data_Trees$pdays[Bank_Data_Trees$pdays==1] <- "NoContact"

# divide to train and test sets

set.seed(123)
Rows_Numbers_for_Test_Set <- sample(1:nrow(Bank_Data),0.2*nrow(Bank_Data),replace = FALSE)

Train_Set_Tree <- Bank_Data_Trees[-Rows_Numbers_for_Test_Set,]
Test_Set_Tree <- Bank_Data_Trees[Rows_Numbers_for_Test_Set,]

# 1

set.seed(123)
tree <- rpart(formula=Train_Set_Tree$y~, data = Train_Set_Tree, method = 'class',
parm=list(split='information'),cp=0)

preds.tree.train <- predict(tree, newdata = Train_Set_Tree[,1:13], type='class')
preds.tree.test <- predict(tree, newdata = Test_Set_Tree[,1:13], type='class')
tree_default_train_precision <- (sum(preds.tree.train==Train_Set_Tree$y))/nrow(Train_Set_Tree)
tree_default_test_precision <- (sum(preds.tree.test==Test_Set_Tree$y))/nrow(Test_Set_Tree)

plot(tree)
text(tree)

# 2

printcp(tree)

plotcp(tree)

opt <- which.min(tree$cptable[, "xerror"]) # get the index of complexity parameter (cp) with lowest xerror
cp <- tree$cptable[opt, "CP"] # get its value

```

```

pruned_model <- prune(tree,cp)

preds.tree.train <- predict(pruned_model, newdata = Train_Set_Tree[,1:13], type='class')

preds.tree.test <- predict(pruned_model, newdata = Test_Set_Tree[,1:13], type='class')

tree_default_train_precision <- (sum(preds.tree.train==Train_Set_Tree$y))/nrow(Train_Set_Tree)

tree_default_test_precision <- (sum(preds.tree.test==Test_Set_Tree$y))/nrow(Test_Set_Tree)

plot(pruned_model)

text(pruned_model)

rpart.plot(pruned_model)

#rpart.plot(pruned_model, extra = 104, box.palette = "GnBu", branch.lty=3, shadow.col = "gray", nn=TRUE)

# just for visualization

pruned_model2 <- prune(tree,0.02)

rpart.plot(pruned_model2)

# 3

#cp=0

pruned_model_cp0 <- prune(tree,0)

preds.tree.train <- predict(pruned_model_cp0, newdata = Train_Set_Tree[,1:13], type='class')

preds.tree.test <- predict(pruned_model_cp0, newdata = Test_Set_Tree[,1:13], type='class')

pruned_model_cp0_train_precision <- (sum(preds.tree.train==Train_Set_Tree$y))/nrow(Train_Set_Tree)

pruned_model_cp0_test_precision <- (sum(preds.tree.test==Test_Set_Tree$y))/nrow(Test_Set_Tree)

#cp with maximal validation error

cp_index<- which.max(tree$cptable[, "xerror"])

cp_maximal_validation_error <- tree$cptable[cp_index, "CP"]

pruned_model_maximal_valid_error_cp <- prune(tree,cp_maximal_validation_error)

preds.tree.train <- predict(pruned_model_maximal_valid_error_cp, newdata = Train_Set_Tree[,1:13], type='class')

```

```

preds.tree.test <- predict(pruned_model_maximal_valid_error_cp, newdata = Test_Set_Tree[,1:13], type='class')

pruned_model_cp_maximal_validation_error_train_precision <-
(sum(preds.tree.train==Train_Set_Tree$y))/nrow(Train_Set_Tree)

pruned_model_cp_maximal_validation_error_test_precision <-
(sum(preds.tree.test==Test_Set_Tree$y))/nrow(Test_Set_Tree)

# cp second best

cp_second_best <- 0.00411946

pruned_model_cp_second_best <- prune(tree,cp_second_best)

preds.tree.train <- predict(pruned_model_cp_second_best, newdata = Train_Set_Tree[,1:13], type='class')

preds.tree.test <- predict(pruned_model_cp_second_best, newdata = Test_Set_Tree[,1:13], type='class')

pruned_model_cp_second_best_train_precision <- (sum(preds.tree.train==Train_Set_Tree$y))/nrow(Train_Set_Tree)

pruned_model_cp_second_best_test_precision <- (sum(preds.tree.test==Test_Set_Tree$y))/nrow(Test_Set_Tree)

# cp 2 splits

pruned_model_2_splits <- prune(tree,0.015105)

preds.tree.train <- predict(pruned_model_2_splits, newdata = Train_Set_Tree[,1:13], type='class')

preds.tree.test <- predict(pruned_model_2_splits, newdata = Test_Set_Tree[,1:13], type='class')

pruned_model_2_splits_train_precision <- (sum(preds.tree.train==Train_Set_Tree$y))/nrow(Train_Set_Tree)

pruned_model_2_splits_test_precision <- (sum(preds.tree.test==Test_Set_Tree$y))/nrow(Test_Set_Tree)

#-----
# K-Means

#-----

# 1

Bank_Data_K_Means <- Train_Set[,-23]

```

```

Train_Set_Original <- Bank_Data_Original[-Rows_Numbers_for_Test_Set,]

Test_Set_Original <- Bank_Data_Original[Rows_Numbers_for_Test_Set,]

# 2

k <- 2

set.seed(123)

clust_data <- kmeans(Bank_Data_K_Means, centers=k)

Bank_Data_K_Means$clust <- factor(clust_data$cluster)

Bank_Data_K_Means$clust_0_1 <- ifelse(Bank_Data_K_Means$clust==1,0,1)

k_means_default_train_percision <- sum(Bank_Data_K_Means$clust_0_1==Train_Set$y)/nrow(Train_Set)

# 3

Bank_Data_K_Means <- Bank_Data_K_Means[,-24]

Bank_Data_K_Means <- data.matrix(Bank_Data_K_Means)

scatt_data <- cls.scatt.data((Bank_Data_K_Means), clust=clust_data$cluster, dist='euclidean')

dunn_train <- clv.Dunn(scatt_data, 'centroid', 'centroid')

DB_train <- clv.Davies.Bouldin(scatt_data, 'centroid', 'centroid')

p1 <- ggplot(Train_Set_Original ,aes(x= Train_Set_Original$age, y=Train_Set_Original$day,
color =ifelse(clust_data$cluster=="1","Green","Red") ,size=10 )) + geom_point(shape =
ifelse(Train_Set_Original$y==1,"Y","N")) + guides(color=F, size=F)

grid.arrange(p1)

```

```
#plots
```

```
Bank_Data_K_Means <- as.data.frame(Bank_Data_K_Means)
```

```
# education
```

```
p1 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$education, y=Train_Set_Original$age, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)
```

```
p2 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$education, y=Train_Set_Original$poutcome , color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)
```

```
p3 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$education, y=Train_Set_Original$housing, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)
```

```
p4 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$education, y=Train_Set_Original$balance, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)
```

```
p5 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$education, y=Train_Set_Original$month, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)
```

```
p6 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$education, y=Train_Set_Original$pdays, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)
```

```
grid.arrange(p1, p2, p3, p4, p5, p6, ncol=3)
```

```
p7 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$education, y=Train_Set_Original$job, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)
```

```
p8 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$education, y=Train_Set_Original$marital , color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)
```

```
p9 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$education, y=Train_Set_Original$loan, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)
```

```
p10 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$education, y=Train_Set_Original$day, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)
```

```
p11 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$education, y=Train_Set_Original$campaign, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)
```

```
p12 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$education, y=Train_Set_Original$previous, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)
```

```
grid.arrange(p7, p8, p9, p10, p11, p12, ncol=3)
```

```
#pdays
```

```
p1 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$pdays, y=Train_Set_Original$age, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)
```

```

p2 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$pdays, y=Train_Set_Original$poutcome , color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p3 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$pdays, y=Train_Set_Original$housing, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p4 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$pdays, y=Train_Set_Original$balance, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p5 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$pdays, y=Train_Set_Original$month, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p6 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$pdays, y=Train_Set_Original$education, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

grid.arrange(p1, p2, p3, p4, p5, p6, ncol=3)

```

```

p7 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$pdays, y=Train_Set_Original$job, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p8 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$pdays, y=Train_Set_Original$marital , color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p9 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$pdays, y=Train_Set_Original$loan, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p10 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$pdays, y=Train_Set_Original$day, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p11 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$pdays, y=Train_Set_Original$campaign, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p12 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$pdays, y=Train_Set_Original$previous, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

grid.arrange(p7, p8, p9, p10, p11, p12, ncol=3)

```

#age

```

p1 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$age, y=Train_Set_Original$pdays, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p2 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$age, y=Train_Set_Original$poutcome , color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p3 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$age, y=Train_Set_Original$housing, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p4 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$age, y=Train_Set_Original$balance, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p5 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$age, y=Train_Set_Original$month, color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

```

```

p6 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$age, y=Train_Set_Original$education,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

grid.arrange(p1, p2, p3, p4, p5, p6, ncol=3)

p7 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$age, y=Train_Set_Original$job,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p8 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$age, y=Train_Set_Original$marital ,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p9 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$age, y=Train_Set_Original$loan,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p10 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$age, y=Train_Set_Original$day,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p11 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$age, y=Train_Set_Original$campaign,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p12 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$age, y=Train_Set_Original$previous,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

grid.arrange(p7, p8, p9, p10, p11, p12, ncol=3)

```

#job

```

p1 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$job, y=Train_Set_Original$pdays,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p2 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$job, y=Train_Set_Original$poutcome ,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p3 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$job, y=Train_Set_Original$housing,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p4 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$job, y=Train_Set_Original$balance,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p5 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$job, y=Train_Set_Original$month,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p6 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$job, y=Train_Set_Original$education,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

grid.arrange(p1, p2, p3, p4, p5, p6, ncol=3)

```

```

p7 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$job, y=Train_Set_Original$age,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p8 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$job, y=Train_Set_Original$marital ,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p9 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$job, y=Train_Set_Original$loan,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

```

```

p10 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$job, y=Train_Set_Original$day,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p11 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$job, y=Train_Set_Original$campaign,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p12 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$job, y=Train_Set_Original$previous,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

grid.arrange(p7, p8, p9, p10, p11, p12, ncol=3)

```

#marital

```

p1 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$marital, y=Train_Set_Original$pdays,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p2 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$marital, y=Train_Set_Original$poutcome ,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p3 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$marital, y=Train_Set_Original$housing,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p4 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$marital, y=Train_Set_Original$balance,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p5 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$marital, y=Train_Set_Original$month,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p6 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$marital, y=Train_Set_Original$education,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

grid.arrange(p1, p2, p3, p4, p5, p6, ncol=3)

```

```

p7 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$marital, y=Train_Set_Original$age,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p8 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$marital, y=Train_Set_Original$job ,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p9 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$marital, y=Train_Set_Original$loan,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p10 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$marital, y=Train_Set_Original$day,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p11 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$marital, y=Train_Set_Original$campaign,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p12 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$marital, y=Train_Set_Original$previous,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

grid.arrange(p7, p8, p9, p10, p11, p12, ncol=3)

```

```
#balance
```

```
p1 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$balance, y=Train_Set_Original$pdays,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p2 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$balance, y=Train_Set_Original$poutcome ,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p3 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$balance, y=Train_Set_Original$housing,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p4 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$balance, y=Train_Set_Original$marital,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p5 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$balance, y=Train_Set_Original$month,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p6 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$balance, y=Train_Set_Original$education,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
grid.arrange(p1, p2, p3, p4, p5, p6, ncol=3)
```

```
p7 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$balance, y=Train_Set_Original$age,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p8 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$balance, y=Train_Set_Original$job ,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p9 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$balance, y=Train_Set_Original$loan,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p10 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$balance, y=Train_Set_Original$day,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p11 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$balance, y=Train_Set_Original$campaign,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p12 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$balance, y=Train_Set_Original$previous,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
grid.arrange(p7, p8, p9, p10, p11, p12, ncol=3)
```

```
#housing
```

```
p1 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$housing, y=Train_Set_Original$pdays,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p2 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$housing, y=Train_Set_Original$poutcome ,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p3 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$housing, y=Train_Set_Original$balance,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p4 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$housing, y=Train_Set_Original$marital,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)
```

```

p5 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$housing, y=Train_Set_Original$month,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p6 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$housing, y=Train_Set_Original$education,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

grid.arrange(p1, p2, p3, p4, p5, p6, ncol=3)

p7 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$housing, y=Train_Set_Original$age,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p8 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$housing, y=Train_Set_Original$job ,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p9 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$housing, y=Train_Set_Original$loan,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p10 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$housing, y=Train_Set_Original$day,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p11 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$housing, y=Train_Set_Original$campaign,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p12 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$housing, y=Train_Set_Original$previous,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

grid.arrange(p7, p8, p9, p10, p11, p12, ncol=3)

```

#loan

```

p1 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$loan, y=Train_Set_Original$pdays,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p2 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$loan, y=Train_Set_Original$poutcome ,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p3 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$loan, y=Train_Set_Original$balance,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p4 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$loan, y=Train_Set_Original$marital,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p5 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$loan, y=Train_Set_Original$month,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p6 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$loan, y=Train_Set_Original$education,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

grid.arrange(p1, p2, p3, p4, p5, p6, ncol=3)

```

```

p7 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$loan, y=Train_Set_Original$age,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p8 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$loan, y=Train_Set_Original$job ,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

```

```

p9 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$loan, y=Train_Set_Original$housing,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p10 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$loan, y=Train_Set_Original$day,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p11 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$loan, y=Train_Set_Original$campaign,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p12 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$loan, y=Train_Set_Original$previous,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

grid.arrange(p7, p8, p9, p10, p11, p12, ncol=3)

```

#day

```

p1 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$day, y=Train_Set_Original$pdays,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p2 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$day, y=Train_Set_Original$poutcome ,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p3 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$day, y=Train_Set_Original$balance,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p4 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$day, y=Train_Set_Original$marital,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p5 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$day, y=Train_Set_Original$month,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p6 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$day, y=Train_Set_Original$education,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

grid.arrange(p1, p2, p3, p4, p5, p6, ncol=3)

```

```

p7 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$day, y=Train_Set_Original$age,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p8 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$day, y=Train_Set_Original$job ,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p9 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$day, y=Train_Set_Original$housing,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p10 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$day, y=Train_Set_Original$loan,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p11 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$day, y=Train_Set_Original$campaign,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p12 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$day, y=Train_Set_Original$previous,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

grid.arrange(p7, p8, p9, p10, p11, p12, ncol=3)

```

```
#month
```

```
p1 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$month, y=Train_Set_Original$pdays,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p2 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$month, y=Train_Set_Original$poutcome ,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p3 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$month, y=Train_Set_Original$balance,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p4 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$month, y=Train_Set_Original$marital,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p5 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$month, y=Train_Set_Original$day,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p6 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$month, y=Train_Set_Original$education,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
grid.arrange(p1, p2, p3, p4, p5, p6, ncol=3)
```

```
p7 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$month, y=Train_Set_Original$age,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p8 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$month, y=Train_Set_Original$job ,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p9 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$month, y=Train_Set_Original$housing,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p10 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$month, y=Train_Set_Original$loan,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p11 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$month, y=Train_Set_Original$campaign,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p12 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$month, y=Train_Set_Original$previous,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
grid.arrange(p7, p8, p9, p10, p11, p12, ncol=3)
```

```
#campaign
```

```
p1 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$campaign, y=Train_Set_Original$pdays,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)  
  
p2 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$campaign, y=Train_Set_Original$poutcome ,  
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)
```

```

p3 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$campaign, y=Train_Set_Original$balance,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p4 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$campaign, y=Train_Set_Original$marital,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p5 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$campaign, y=Train_Set_Original$day,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p6 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$campaign, y=Train_Set_Original$education,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

grid.arrange(p1, p2, p3, p4, p5, p6, ncol=3)

```

```

p7 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$campaign, y=Train_Set_Original$age,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p8 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$campaign, y=Train_Set_Original$job ,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p9 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$campaign, y=Train_Set_Original$housing,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p10 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$campaign, y=Train_Set_Original$loan,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p11 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$campaign, y=Train_Set_Original$month,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p12 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$campaign, y=Train_Set_Original$previous,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

grid.arrange(p7, p8, p9, p10, p11, p12, ncol=3)

```

#previous

```

p1 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$previous, y=Train_Set_Original$pdays,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p2 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$previous, y=Train_Set_Original$poutcome ,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p3 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$previous, y=Train_Set_Original$balance,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p4 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$previous, y=Train_Set_Original$marital,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p5 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$previous, y=Train_Set_Original$day,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p6 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$previous, y=Train_Set_Original$education,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

grid.arrange(p1, p2, p3, p4, p5, p6, ncol=3)

```

```

p7 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$previous, y=Train_Set_Original$age,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p8 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$previous, y=Train_Set_Original$job ,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p9 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$previous, y=Train_Set_Original$housing,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p10 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$previous, y=Train_Set_Original$loan,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p11 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$previous, y=Train_Set_Original$month,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p12 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$previous, y=Train_Set_Original$campaign,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

grid.arrange(p7, p8, p9, p10, p11, p12, ncol=3)

```

#poutcome

```

p1 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$poutcome, y=Train_Set_Original$pdays,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p2 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$poutcome, y=Train_Set_Original$previous ,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p3 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$poutcome, y=Train_Set_Original$balance,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p4 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$poutcome, y=Train_Set_Original$marital,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p5 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$poutcome, y=Train_Set_Original$day,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p6 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$poutcome, y=Train_Set_Original$education,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

grid.arrange(p1, p2, p3, p4, p5, p6, ncol=3)

```

```

p7 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$poutcome, y=Train_Set_Original$age,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p8 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$poutcome, y=Train_Set_Original$job ,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p9 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$poutcome, y=Train_Set_Original$housing,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p10 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$poutcome, y=Train_Set_Original$loan,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

```

```

p11 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$poutcome, y=Train_Set_Original$month,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

p12 <- ggplot(Bank_Data_K_Means, aes(x=Train_Set_Original$poutcome, y=Train_Set_Original$campaign,
color=clust_data$cluster, size=10)) + geom_point() + guides(color=F, size=F)

grid.arrange(p7, p8, p9, p10, p11, p12, ncol=3)

```

FINDING THE BEST K

```
Bank_Data_K_Means <- Train_Set[,-23]
```

```
Bank_Data_K_Means <- data.matrix(Bank_Data_K_Means)
```

```
# Dunn & DB
```

```
set.seed(123)
```

```
dunn <- c(); DB <- c(); K <- 9
```

```
for(k in 2:K){
```

```
  clust_data <- kmeans(Bank_Data_K_Means, centers=k)
```

```
  scatt_data <- cls.scatt.data(Bank_Data_K_Means, clust=clust_data$cluster, dist='euclidean')
```

```
  dunn <- c(dunn, clv.Dunn(scatt_data, 'centroid', 'centroid'))
```

```
  DB <- c(DB, clv.Davies.Bouldin(scatt_data, 'centroid', 'centroid'))
```

```
}
```

```
clust_metrics <- data.frame(K = rep(seq(2,K,1),2), value = c(dunn, DB), metric = c(rep('Dunn',K-1), rep('DB',K-1)))
```

```
ggplot(clust_metrics, aes(x=K, y=value, color=factor(metric))) + geom_point() + geom_line()
```

```
k <- c(2,3,4,5,6,7,8,9)
```

```
print(cbind(k,DB,dunn, DB-dunn))
```

```
# kmeans results
```

```
clust_data_2 <- kcca(Bank_Data_K_Means[,-23], k=2, kccaFamily("kmeans"))

pred_train <- predict(clust_data_2)

pred_test <- predict(clust_data_2, newdata=data.matrix(Test_Set[,-23]))

pred_test <- pred_test-1
```

```
### Result for test ---#
```

```
k_means_default_test_percision <- sum(pred_test==Test_Set$y)/nrow(Test_Set)
```

```
#-----
```

```
# SVM
```

```
#-----
```

```
## classification mode
```

```
# default with factor response:
```

```
model_svm <- svm(as.factor(Train_Set$y) ~ ., data = Train_Set)
```

```
print(model_svm)
```

```
summary(model_svm)
```

```
# train perciosion
```

```
pred_train_svm <- predict(model_svm, Train_Set[,1:22])

pred_train_svm <- c(pred_train_svm)

pred_train_svm <- pred_train_svm-1

svm_train_perciosin <- sum(pred_train_svm==Train_Set$y)/nrow(Train_Set)
```

```
# test perciosion

pred_test_svm <- predict(model_svm, Test_Set[,1:22])

pred_test_svm <- c(pred_test_svm)

pred_test_svm <- pred_test_svm-1

svm_test_perciosin <- sum(pred_test_svm==Test_Set$y)/nrow(Test_Set)
```

confusion table:

```
table(pred_test_svm, Test_Set$y)
```

```
#-----
```

Predictions for X-Test

```
#-----
```

data preperation

```
filePath=choose.files()

X_Test_Original<-read.csv(filePath,header=TRUE)
```

```
X_Test <- X_Test_Original
```

delete variables

```
delete <- c(5,9,16)

X_Test <- X_Test[,-delete]
```

```
# complete unknown data
```

```

X_Test[X_Test=="unknown"] <- NA

X_Test <- mice(data=X_Test, m=5, method="pmm", maxit=50, seed=50)

X_Test <- complete(X_Test)

# month to quarters

X_Test$month <- as.character(X_Test$month)

X_Test$month[X_Test$month=="jan" | X_Test$month=="feb" | X_Test$month=="mar"] <- 1
X_Test$month[X_Test$month=="apr" | X_Test$month=="may" | X_Test$month=="jun"] <- 2
X_Test$month[X_Test$month=="jul" | X_Test$month=="aug" | X_Test$month=="sep"] <- 3
X_Test$month[X_Test$month=="oct" | X_Test$month=="nov" | X_Test$month=="dec"] <- 4

X_Test$month <- as.factor(X_Test$month)

# pdays to categories

X_Test$pdays <- findInterval(X_Test$pdays,c(-1,0,181,366,1000))

X_Test$pdays[X_Test$pdays==4] <- "OverOneYear"
X_Test$pdays[X_Test$pdays==3] <- "OneYear"
X_Test$pdays[X_Test$pdays==2] <- "HalfYear"
X_Test$pdays[X_Test$pdays==1] <- "NoContact"

# Dummy variables

#pdays

X_Test$pdays_D1 <- ifelse(X_Test$pdays=="HalfYear",1,-1)
X_Test$pdays_D2 <- ifelse(X_Test$pdays=="OneYear",1,-1)
X_Test$pdays_D3 <- ifelse(X_Test$pdays=="OverOneYear",1,-1)

X_Test$pdays_D1 <- as.factor(X_Test$pdays_D1)
X_Test$pdays_D2 <- as.factor(X_Test$pdays_D2)
X_Test$pdays_D3 <- as.factor(X_Test$pdays_D3)

```

```
# job  
X_Test$job_D1 <- ifelse(X_Test$job=="blue-collar",1,-1)  
X_Test$job_D2 <- ifelse(X_Test$job=="management",1,-1)  
X_Test$job_D3 <- ifelse(X_Test$job=="technician",1,-1)
```

```
X_Test$job_D1 <- as.factor(X_Test$job_D1)  
X_Test$job_D2 <- as.factor(X_Test$job_D2)  
X_Test$job_D3 <- as.factor(X_Test$job_D3)
```

```
#marital
```

```
X_Test$marital_D1 <- ifelse(X_Test$marital=="married",1,-1)  
X_Test$marital_D2 <- ifelse(X_Test$marital=="divorced",1,-1)
```

```
X_Test$marital_D1 <- as.factor(X_Test$marital_D1)  
X_Test$marital_D2 <- as.factor(X_Test$marital_D2)
```

```
#education
```

```
X_Test$education_D1 <- ifelse(X_Test$education=="secondary",1,-1)  
X_Test$education_D2 <- ifelse(X_Test$education=="tertiary",1,-1)
```

```
X_Test$education_D1 <- as.factor(X_Test$education_D1)  
X_Test$education_D2 <- as.factor(X_Test$education_D2)
```

```
#month
```

```
X_Test$Quarter_D1 <- ifelse(X_Test$month=="2",1,-1)  
X_Test$Quarter_D2 <- ifelse(X_Test$month=="3",1,-1)  
X_Test$Quarter_D3 <- ifelse(X_Test$month=="4",1,-1)
```

```
X_Test$Quarter_D1 <- as.factor(X_Test$Quarter_D1)  
X_Test$Quarter_D2 <- as.factor(X_Test$Quarter_D2)  
X_Test$Quarter_D3 <- as.factor(X_Test$Quarter_D3)
```

```
#poutcome
X_Test$poutcome_D1 <- ifelse(X_Test$poutcome=="failure",1,-1)
X_Test$poutcome_D2 <- ifelse(X_Test$poutcome=="other",1,-1)

X_Test$poutcome_D1 <- as.factor(X_Test$poutcome_D1)
X_Test$poutcome_D2 <- as.factor(X_Test$poutcome_D2)

#housing
X_Test$housing_D1 <- ifelse(X_Test$housing=="yes",1,-1)

X_Test$housing_D1 <- as.factor(X_Test$housing_D1)

#loan
X_Test$loan_D1 <- ifelse(X_Test$loan=="yes",1,-1)

X_Test$loan_D1 <- as.factor(X_Test$loan_D1)

# columns deletion
deletion <- c(2,3,4,6,7,9,11,13)
X_Test <- X_Test[,-deletion]

# scaling
X_Test$age <- scale(X_Test$age)
X_Test$balance <- scale(X_Test$balance)
X_Test$day <- scale(X_Test$day)
X_Test$campaign <- scale(X_Test$campaign)
X_Test$previous <- scale(X_Test$previous)
```

```
#-----  
  
# prediction  
  
preds_X_Test <- factor(predict(nn_chosen, newdata=X_Test, type='class'))  
Results <- as.data.frame(preds_X_Test)  
  
  
path=choose.files()  
write_xlsx(x = Results, path = path, col_names = TRUE)
```