מסמך פרויקט חשיבה וקבלת החלטות

חקר ראשוני

תחילה התחלנו מלממש רגרסיה לוגסיטית. עשינו את זה בתור בייסליין כדי להבין מה התחזיות שאפשר לקבל על המודל הכי פשטני שיש. תחילה לקחנו את מאגר המידע הכולל את כל הניסויים של האנשים. הכנסנו אותו בשני צורות למודל, ורצינו לראות מה התחזיות שמקבלים. שני הצורות היו: 1. להכניס רק בעיה ספציפית (בחרנו בעיה 11) 2. להכניס את הכל, ולבדוק תחזיות לפי הכל. קיבלנו את התוצאות הבאות:

```
from sklearn.metrics import accuracy_score
import numpy as np

accuracy = accuracy_score(y_test, y_pred)
print(f"predicted probability to choose A in task 11 is: {np.mean(y_pred)}")
```

```
predicted probability to choose A in task 11 is: 1.0
```

```
from sklearn.metrics import accuracy_score
import numpy as np

accuracy = accuracy_score(y_test, y_pred)
print(f"predicted probability to choose A over all tasks is: {np.mean(y_pred)}")
```

```
predicted probability to choose A over all tasks is: 0.6113947173537398
```

מכאן קיבלנו שני תוצאות מעניינות - א. רגרסיה לוגיסטית חוזה על בעיה 11 שתמיד בוחרים בA אם מתאמנים רק על בעיה 11. ב. רגרסיה לוגיסטית חוזה יותר טוב אם מאמנים אותה על כל הבעיות.
החלטנו לבדוק האם מודל יותר מורכב מרגרסיה לוגיסטית יוכל להשיג תוצאות טובות יותר.
הגענו למסקנה שיש שני מודלים המתאימים במיוחד לבעיות מסוג זה:
1. עצי החלטה - ההתאמה פה ברורה - אנחנו מתעסקים בהחלטות של אנשים, ולכן נראה סביר כי עצי החלטה יהוו מודל מתאים
2. מודל הPAS - המודל שנלמד בכיתה וראינו כי ממדל קבלת החלטות של אנשים ולכן סביר שייתן תוצאה טובה.
כעת המשימה הינה לממש את שני המודלים, ולבדוק מה נותן תחזיות יותר טובות.

עצי החלטה

הכנסנו את הנתונים לעץ ההחלטה, וקיבלנו תחזיות הרבה יותר טובות (MSE יחסית נמוך של 2.6) אך זה התוצאות רק על סמך הפיצ'רים שנתונים מראש (a1,pa1,a2 etc). כעת, על מנת לשדרג את המודל הוספנו פיצ'רים נוספים לדוג ע"י ביצוע פעולות אריתמטיות בין המשתנים - לדוגמה: חיבור של a1 וa2, חיסור בין התוחלות של a ושל b. הכפלה של a1 וb1, מקסימום בין a1 לa2, מינימום של הערכים (נזק מקסימלי), כל אלו מהווים שיקולים אפשריים של אנשים בעת קבלת ההחלטה על איזה כפתור ללחוץ. הכנסנו את כל זה למודל, וקיבלנו תוצאה יותר טובה (התמונה הימנית)

```python
df[df["a2"] == "."] = 0
df[df["b2"] == "."] = 0
df["muA"] = df["a1"]*df["pa1"] + df["a2"]*(1-df["pa1"])
df["muB"] = df["b1"]*df["pb1"] + df["b2"]*(1-df["pb1"])
df["varA"] = np.sqrt((df["a1"]**2) * df["pa1"] + (df["a2"]**2) * (1-df["pa1"]))
df["varB"] = np.sqrt((df["b1"]**2) * df["pb1"] + (df["b2"]**2) * (1-df["pb1"]))
df["dif_mu"] = df["muA"] - df["muB"]
df["dif_1"] = df["a1"] - df["b1"]
df["dif_2"] = df["a2"] - df["b2"]
df["dif_a"] = df["a1"] - df["a2"]
df["dif_b"] = df["b1"] - df["b2"]
#df["abs_dif_mu"] = df["dif_mu"].abs()
df["mul_mu"] = df["muA"] * df["muB"]
df["mul_1"] = df["a1"] * df["b1"]
df["mul_2"] = df["a2"] * df["b2"]
df["mul_a"] = df["a1"] * df["a2"]
df["mul_b"] = df["b1"] * df["b2"]
df["add_mu"] = df["muA"] + df["muB"]
df["max1"] = df[["a1", "b1"]].max(axis=1)
df["max2"] = df[["a2", "b2"]].max(axis=1)
df["max_damage"] = df[["a1","a2", "b1", "b2"]].min(axis=1)
df = df.fillna(0)
mse = []
pred1 = []
pred2 = []
for i in range(1000):
  X_train, X_test, y_train, y_test, x_val = prepare(df)
  dtr = DecisionTreeRegressor(random_state=42)
  dtr.fit(X_train, y_train)
  y_pred = dtr.predict(X_test)
  mse_i = mean_squared_error(y_test, y_pred)
  mse.append(mse_i)
  val = dtr.predict(x_val)
  pred1.append(val[0])
  pred2.append(val[1])
print(f"MSE = {np.mean(mse)}\npred1 = {np.mean(pred1)}\npred2={np.mean(pred2)}")
```

```
MSE = 0.032376072
pred1 = 0.6475299999999999
pred2=0.5183399999999999
```

```python
df[df["a2"] == "."] = 0
df[df["b2"] == "."] = 0
df["muA"] = df["a1"]*df["pa1"] + df["a2"]*(1-df["pa1"])
df["muB"] = df["b1"]*df["pb1"] + df["b2"]*(1-df["pb1"])
df["varA"] = np.sqrt((df["a1"]**2) * df["pa1"] + (df["a2"]**2) * (1-df["pa1"]))
df["varB"] = np.sqrt((df["b1"]**2) * df["pb1"] + (df["b2"]**2) * (1-df["pb1"]))
df["dif_mu"] = df["muA"] - df["muB"]
df["dif_1"] = df["a1"] - df["b1"]
df["dif_2"] = df["a2"] - df["b2"]
df["dif_a"] = df["a1"] - df["a2"]
df["dif_b"] = df["b1"] - df["b2"]
df["abs_dif_mu"] = df["dif_mu"].abs()
df["mul_mu"] = df["muA"] * df["muB"]
df["mul_1"] = df["a1"] * df["b1"]
df["mul_2"] = df["a2"] * df["b2"]
df["mul_a"] = df["a1"] * df["a2"]
df["mul_b"] = df["b1"] * df["b2"]
df["add_mu"] = df["muA"] + df["muB"]
df["max1"] = df[["a1", "b1"]].max(axis=1)
df["max2"] = df[["a2", "b2"]].max(axis=1)
df["max_damage"] = df[["a1","a2", "b1", "b2"]].min(axis=1)
df = df.fillna(0)
mse = []
pred1 = []
pred2 = []
for i in range(1000):
  X_train, X_test, y_train, y_test, x_val = prepare(df)
  dtr = DecisionTreeRegressor(random_state=42)
  dtr.fit(X_train, y_train)
  y_pred = dtr.predict(X_test)
  mse_i = mean_squared_error(y_test, y_pred)
  mse.append(mse_i)
  val = dtr.predict(x_val)
  pred1.append(val[0])
  pred2.append(val[1])
print(f"MSE = {np.mean(mse)}\npred1 = {np.mean(pred1)}\npred2={np.mean(pred2)}")
```

```
MSE = 0.031931288
pred1 = 0.6517200000000001
pred2=0.52041
```

כאשר תוצאות האמת הינן 0.62 ו0.62. מכאן שאנו מגיעים לתוצאות הרבה יותר קרובות למה שאמור לקרות - כלומר אנו משפרים את המודל.

לבסוף אנחנו היינו צריכים לבדוק אילו מהפיצרים הינם חשובים, ואילו לא. בשביל זה התחלנו להוריד מהפיצרים ולבדוק את השינוי של התוצאות (התמונה השמאלית).

אך הגענו לבעיה שזה גם איטי מדי, וגם לא נכון סטטיסטית. אז הרצנו קוד של sci-kit שעושה הערכה לחשיבות של הפיצ'רים. מה שעשינו היה להריץ את הקוד 10000 פעמים, וכל פעם להשתמש בפקודה של sci-kit כדי לקחת את ה12 הכי חשובים:

```
MSE = 0.029911059599999998
pred1 = 0.67706
pred2 = 0.518608
sorted dict for Arate1 is
{8: 10000, 33: 9998, 32: 9937, 1: 9802, 11: 9633, 5: 9626, 15: 9002, 29: 8822, 9: 8025, 0: 7401, 4: 5464, 31: 5063
best options are: Index(['Est_A0', 'maxmu', 'max2', 'pa1', 'SDA', 'b2', 'dif_2', 'add_a', 'muA',
       'a1', 'pb1', 'max1'],
      dtype='object')
```

מכאן לקחנו את כל הפיצ'רים שמספר הפעמים שהם הופיעו הינה מעל 9000, והשתמשנו בהם בשביל להריץ את הפרדיקציות של Arate1,2... קיבלנו את הפרדיקציות הבאות (עבור Arate1 - Arate4 משמאל לימין בהתאמה)

```
MSE = 0.0281872976          MSE = 0.05464974520000001  MSE = 0.0485715899          MSE = 0.043781825999999996
pred1 = 0.5751430000000001  pred1 = 0.6832539999999999 pred1 = 0.671429            pred1 = 0.665322
pred2 = 0.5121330000000001  pred2 = 0.470605           pred2 = 0.5545730000000001  pred2 = 0.503788
```

כעת נותר להשוות לתוצאות של PAS.

מודל PAS

לצורך המימוש לקחנו את הקוד ההפוך ניסויים לפרוספקטים מהמאמר [erev et el. 2023] ואת מודל PAS של DEBM בו נעשה שימוש והרצנו כדי לקבל בייסליין של מודל PAS מייד ראינו שיפור לעמות הבייסליין של מודל הרגרסיה הנאיבי, אבל עדיין ביצועים שלא עומדים מול מודל עצי ההחלטה עם הפיצ'רים הנוספים.
לאחר עיון בתחזיות המודל הגענו למספר מסקנות והתחלנו לבצע איטרציות של שינויים ובדיקות.
אחד היתרונות של מודל לוגי מסביר הוא שניתן להסיק מסקנות באופן יותר ישיר מהתבוננות במשקולות של עצי החלטה כיוון שאתה יכול לתת הסברים לשגיאות המודל (דוגמה בהמשך).

שמנו לב שהמודל המקורי בוחר לדגום באופן שווה מכל ההיסטוריה כאשר הוא מבצע את הsampling הנרמז מפרמטר kappa, אולם אנחנו חושבים כי סביר יותר שאנשים ידגמו באופן מוטה כלפי ההיסטוריה הקרובה, ובפרט הנחנו שאנשים תמיד יזכרו את התוצאה האחרונה, ובנוסף יתנו משקל גדול יותר לדברים האחרונים שקרו. הדרך בה מימשנו את תפיסה זו הוא שכאשר בוחרים idx אז תמיד ניקח את התוצאה האחרונה ובנוסף נדגום באופן מוטה, K⅔ האירועים האחרונים יילקחו בחשבון ובנוסף K⅓ יידגם מכל המדגם באופן שווה (כאשר ניתן לדגום תוצאות מספר פעמים מה שאומר שבהסתברות ניתן משקל גדול יותר לאירועים האחרונים).
דבר זה גרם לבעיה כיוון שעבור K קטן (<5) בגלל הדגימה המוטה אנחנו לא מצליחים לתפוס את המקרים בהם יש תוצאות קיצוניות בהסתברות נמוכה ולכן גם הגדלנו את K באופן ניכר (עשינו מספר ניסיונות והגענו למסקנה שאיזון טוב יהיה K=10).
כחלק מהגדלת K הרגשנו כי דגימה באופן יוניפורמי איננה מתאימה למשימה זו וכי יותר סביר שהתפלגות K בקרב אנשים היא נורמלית דווקא (כיוון שזה משתנה רנדומי ואנחנו מסתכלים על "אוכלסייה" מסומלצת של 1000 אנשים בכל ניסוי).

במימוש PAS של DEBM ההנחה היא כי אין סימון על הכפתורים ולכן הלחיצה הראשונה היא אקראית לחלוטין, כיוון שאצלנו יש סימון והלחיצה הראשונה לא אקראית החלטנו להוסיף למודל את הנתון של ההסתברות ללחיצה הראשונה ולהשתמש בו כדי לדמות את תנאי ההתחלה באופן קרוב יותר למציאות

עם מודל הPAS אנחנו קיבלנו את התוצאות הבאות:

```
Arate1
Mean Squared Error: 0.013900504950495052
Arate2
Mean Squared Error: 0.019231514851485148
Arate3
Mean Squared Error: 0.0184069801980198
Arate4
Mean Squared Error: 0.019743465346534653
```

בחירת מודל

כאשר אנו משווים בין מודל הPAS לבין עצי ההחלטה, אנחנו רואים שהPAS מחזיר MSE נמוך יותר ובנוסף מסביר את ההחלטות, ועל כן אנו בחרנו לחזות על פיו. ואת התחזית צירפנו למייל בנפרד כפי שנדרש.

הערה: השתמשנו בהשוואה בMSE במקום MSD, שכן לא כל כך ניתן לבצע הערכה לפי MSD בשימוש על עצים שכן MSD (להבנתינו) מודד דאטא על גבי זמן, ועצים מחזירים החלטה בשלב 1. מכיוון שהיינו צריכים לבצע השוואה ביניהם, השתמשנו בMSE שניתן למדידה עבור שני המודלים.