УНИВЕРЗИТЕТ „СВЕТИ КИРИЛ И МЕТОДИЈ“ – СКОПЈЕ
ФАКУЛТЕТ ЗА ИНФОРМАТИЧНИ НАУКИ И КОМПЈУТЕРСКО ИНЖИНЕРСТВО

**СТАНДАРДЕН ПРОЕКТ ПО ПРЕДМЕТОТ – ВОВЕД ВО НАУКАТА НА ПОДАТОЦИ**

тема:

# Методи за робусна проценка на хетерогени каузални ефекти

Ментор:
магистер Јована Добрева

Студент:
Драгана Трифунова

Скопје, август 2025.

# СОДРЖИНА

# ВОВЕД

Почитувани,

во оваа документација која го опишува стандардниот проект по предметот „Вовед во науката на податоци“ ќе се занимаваме со опишување на алгоритмите за каузален ефект и нивна меѓусебна споредба.

Методите за хетероген каузален ефект служат за проценка на тоа како третманот влијае различно на различни подгрупи во популацијата.
Во споредба со стандардните методи за машинско учење тие се поспецифични методи кои имаат поинаква примена во секојдневието. Одговараат на прашањата:

*„Зошто некои луѓе реагираат поинаку на лекот/третманот/политиката?“,*
*„Кои групи ќе имаат најголема корист?“,*
*„Како да распределиме ограничени ресурси (пари, лекови, интервенции)?“*

Иако на прв поглед можеби ни изгледа дека овие методи за хетероген каузален ефект имаат примена само во медицината сепак, тоа не е така и тие имаат секојдневна примена и во економијата, маркетингот, секојдневниот живот како и на многу други места за кои ќе зборуваме подоцна.

Во овој извештај најпрвин ќе зборуваме за податочното множество што ќе го користиме па ќе се префрлиме на имплементацијата на Double Machine Learning и Causal Forest како и на нивните предности и негативности а, за крај малку ќе обопштиме и ќе се насочиме кон заклучоците што можат да се извлечат од овој проект.

Со почит,
Драгана Трифунова

## ПОДАТОЧНО МНОЖЕСТВО

Името на податочното множество кое ќе го користиме во овој проект се нарекува: **ACIC Causal Inference Challenge Datasets** и во него ние анализираме карактеристики на пациенти и информации за тоа дали пациентите примиле терапија и кои се исходите од неа.
Двата датасетови кои ги користам во проектот се најдени на github:

https://github.com/BiomedSciAI/causallib/blob/master/causallib/datasets/data/acic_challenge_2016/README.md

и се претставници на ова податочно множество. Се работи за две табели **x.csv** и **zymu_1.csv**.

Во првата табела имаме коваријати односно карактеристики на пациентите. (Вкупно 58 атрибути)

Во втората табела имаме 5 колони. Првата колона **z** ни кажува дали пациентот примил третман или не. Втората и третата колона ни се исходите **y0** и **y1** се потенцијални исходи од каузална анализа, кои ја мерат ефективноста на третманот (**z**) врз одредена променлива од интерес.

y0 ја претставува вредноста на исходната променлива кога субјектот не е изложен на третманот (z = 0).

y1 ја претставува вредноста на исходната променлива кога субјектот е изложен на третманот (z=1).

Во каузалната анализа, само еден од исходите (y0 или y1) е набљудуван во реалните податоци, во зависност од тоа дали субјектот примил третман (z = 1) или не (z = 0). Другиот исход останува недовршен (контрафактуален) и мора да се процени.
Тоа значи дека доколку пациентот примил третман исходот y1 е реален а, исходот y0 е проценет. Или ако пациентот не примил третман го проценуваме исходот што ќе се случи ако го примил третманот а, го

мериме исходот од непримениот третман. Каузалниот ефект се пресметува на следниот начин:

y1-y0 = проценка колку лекот помага

y0 и y1 се потенцијални исходи, но само еден е набљудуван.
Целта на каузалната анализа е да се процени оној што го немаме, за да се мери вистинскиот ефект на третманот.

И последните 2 колони mu0 и mu1 ни соодветствуваат со y0 и y1 со тоа што овие колони ги содржат истите податоци но без шум. (Нема да ги користиме бидеиќи ретко се дадени во пракса)

# CAUSAL FOREST

Causal Forest е метод за каузална инференција кој се базира на идејата на *random forest*, но е специјално дизајниран за проценка на каузални ефекти. Наместо да предвидува вредности или класи, тој се обидува да ја процени разликата помеѓу потенцијалните исходи под третман и под контрола.

Со помош на голем број „дрва на одлука“, Causal Forest гради ансамбл модел кој учи како различни коваријати влијаат на ефектот на третманот. Овој метод овозможува да се откријат **хетерогени каузални ефекти**, односно разлики во влијанието на третманот кај различни групи или индивидуи.

Една од неговите предности е тоа што комбинира флексибилност и робусност, бидејќи ги користи идеите на машинското учење, но е насочен кон каузална анализа.

Causal Forest често се користи во економија, медицина и општествени науки за да се разбере кои подгрупи од населението најмногу (или најмалку) имаат корист од одредена интервенција.

# DOUBLE MASCHINE LEARNING

Double Machine Learning (DML) е метод за каузална инференција кој комбинира техники од економетрија и машинско учење со цел да се добие непристрасна проценка на каузалните ефекти, дури и кога податоците се комплексни и содржат многу коваријати. Главната идеја е дека прво се користат модели од машинско учење за да се „отстранат“ или контролираат влијанијата на коваријатите врз третманот и исходот, а потоа во втора фаза се пресметува ефектот на третманот врз исходот на „прочистените“ податоци. Со тоа се намалува ризикот од пристрасност која може да настане ако директно се примени регресија или друг класичен метод.

Double Machine Learning е особено корисен во ситуации каде што има многу објаснувачки варијабли или каде односите меѓу нив и исходот се нелинеарни и сложени. Овој пристап овозможува истовремено искористување на предностите на современото машинско учење за моделирање на комплексни врски, но и гарантирање на статистичка валидност при процена на каузални ефекти.

*Забелешка: Во овој проект беше планирано да се имплементира и моделот на BART но за жал тој нема стабилна и официјална Python имплементација која е лесно дотапна преку pip install. Многубројните обиди за успешно импортирање не BART за жал завршија без успех.*

## ИМПЛЕМЕНТАЦИЈА

Во оваа секција постепено ќе го објаснам целиот код кој го напишав во овој проект. Најпрвин направив некој основни предпроцесирања на табелите кои ги опишав во претходната секција. Проверив дали има недостасувачки вредности, дали класите се балансирани, дали има категориски променливи и дали има мултиколинеарност помеѓу атрибутите во податочното множество. Како што напоменав порано атрибутите се дадени во една табела а, целната класа (1 = примил третман, 0 = не примил третман) и потенцијалните исходи од каузалната анализа (y0 и y1) се дадени во друга табела. Постепено ќе ги објаснам сите овие чекори:

**Недостасувачки вредности:** Во ова податочно множество ниту во едната ниту во другата табела нема недостасувачки вредности.

**Балансираност на класите:**

```
zymu1['z'].value_counts()
0   3944
1   858
```

*Слика1 : Проверка на балансираност на класата за примање на третманот*

Иако има 4.6 пати повеќе пациенти што не го примиле третманот наспроти тие што го примиле сепак овие размери и не се толку големи за да отстрануваме пациенти од оние кои не се го примиле тратманот. (Слика 1)

**Категориски променливи:**

Во ова податочно множество во првата табела (табелата на коваријати) има 3 категориски променливи. Овде морав да изберам дали ќе користам Label Encoder или пак One-Hot Encoder.
Причината зошто не користам One-Hot Encoder е затоа што ќе ми создаде дополнително околу 26 колони во табелата на коваријати кои ќе ми претставуваат потешкотии за понатамошната визуелизација.

Иако Label Encoder воведува лажен редослед помеѓу категориите одлучив дека во нашиот случај тој е подоброто решение. (слика 2)

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
categorical_cols = ['x_2', 'x_21', 'x_24']
for col in categorical_cols:
x[col] = le.fit_transform(x[col])
```

*Слика2 : Енкодирање на категориски променливи*

Во втората табела немаме категориски променливи.

**Мултиколинеарност помеѓу атрибутите:**

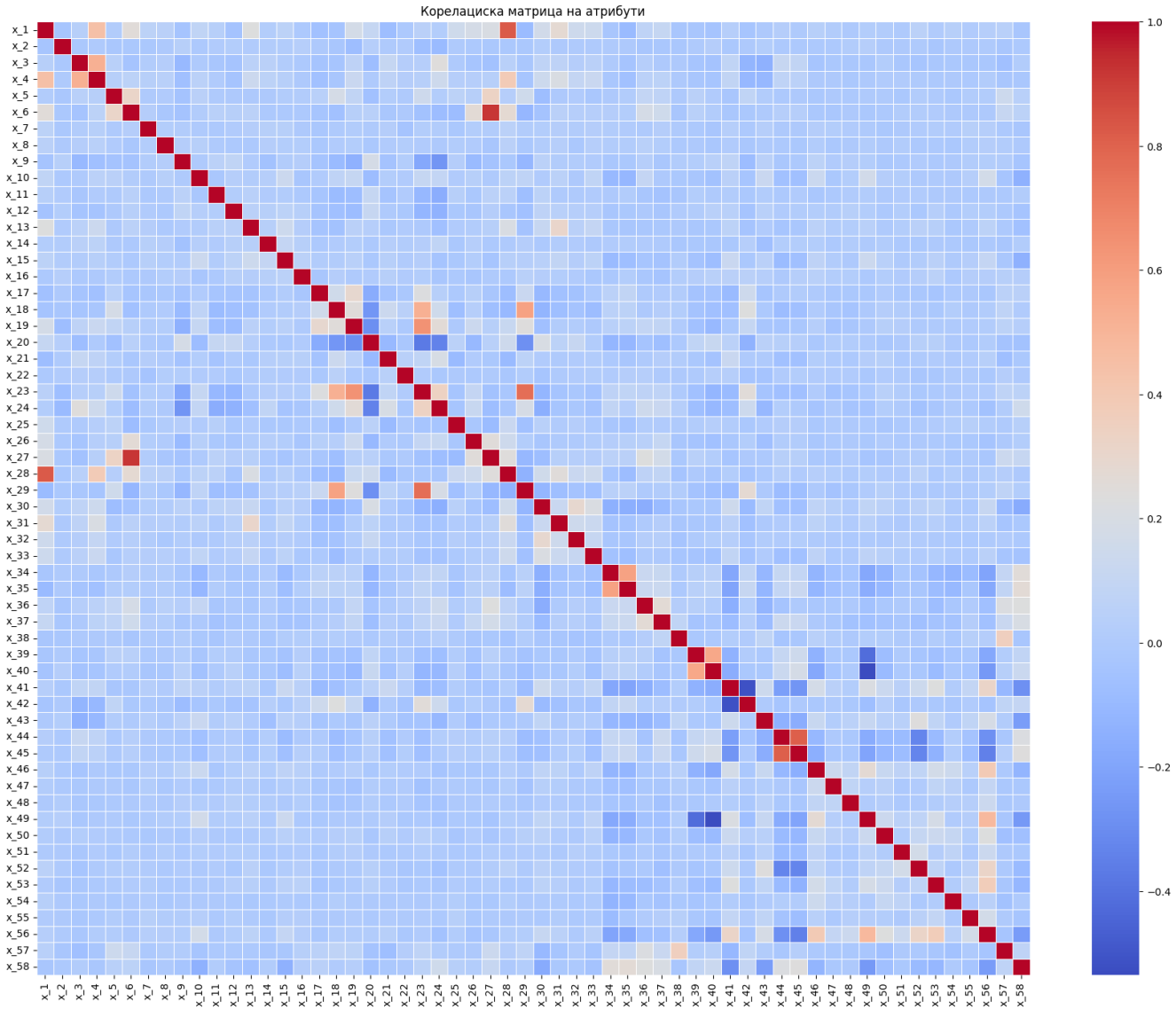
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(22, 17))
correlation_matrix = x.corr()
sns.heatmap(correlation_matrix, annot=False, cmap='coolwarm', linewidths=0.5)
plt.title('Корелациска матрица на атрибути')
plt.show()
```

*Слика3: Код за корелациска мапа на коваријатите*

Треба да провериме дали карактеристиките на пациентите имаат корелација помеѓу себе. Доколку коефициентот на корелација е многу близок до 1 или до -1 значи постојат атрибути кои се многу корелирани. Ова не е убаво од две причини:

1. Може да направат компликации во моделите - overfitting (Понекогаш тоа се решава со PCA)
2. Информацијата што ќе се добие од едниот атрибут можеме да ја добиеме како линеарна комбинација од другиот атрибут и задржувањето на атрибутите во овој случај значи поголема комплексност за моделот.

Затоа креираме код за да нацртаме корелациска мапа за атрибутите и внимателно да ја разгледаме. (слика 3)




*Слика4: Корелациска мапа на коваријатите*

Според легендата на слика 4 можеме да видиме дека доколку полињата се обоени во интензивна црвена боја или пак во интензивна сина боја тогаш тоа значи позитивна односно негативна корелација. Генералната слика на овие коефициенти е добра односно се слабо корелирани.
Сепак, на цртежот можеме да забележиме по некое такво поле на пример полето помеѓу x_1 и x_28.
Поради тоа што матрицата содржи многу атрибути на сликата ние не можеме да го видиме точниот коефициент на корелација помеѓу овие две полиња. Од тие причини направивме код кој ќе ни ги пронајде сите колони

кои имаат корелираност во интервалот од (-0.8, -∞) и од (0.8, ∞) и тој е прикажан на слика 5.

```
threshold = 0.8
high_corr_pairs = []
for col1 in correlation_matrix.columns:
for col2 in correlation_matrix.columns:
if col1 != col2:
corr_value = correlation_matrix.loc[col1, col2]
if abs(corr_value) > threshold:
high_corr_pairs.append((col1, col2, corr_value))
```

*Слика5: Пронаоѓање на сите колони за кои важи дека апсолутната вредност на нивниот коефициент на корелација е поголема од 0.8*

Ги пронаоѓаме овие колони и отстрануваме по една од секој пар соодветно.

```
T = zymu1['z'].values       # dali primil tretman ili ne
y0 = zymu1['y0'].values
y1 = zymu1['y1'].values
Y = np.where(T == 0, y0, y1)
```

*Слика6: Извлекување на важни информации за целната класа*

Следно што треба да го направиме е да ги извлечеме колоните од табелата zymu1. (слика 6) Најпрвин ја извлекуваме информацијата за тоа дали поединецот примил третман или не. Тоа го запишуваме во променливата T. Од оваа променлива зависи информацијата во променливата Y. Затоа што доколку поединецот примил третман тогаш во Y ќе биде вистинската вредност од тој исход y1 а, не претпоставената (y0) и обратно. Ако T е 0 тоа значи дека поединецот не примил третман па, во Y ќе биде вредноста на y0. Исто така имаме уште 2 променливи y0 и y1. Во нив се сместени двата исходи за секој поединец (едниот измерен а, другиот предвиден како што беше опишано погоре). Сега кога веќе ги поделивме колоните од целната класа можеме да започнеме со поделба на целото множество на тренинг и на тест множество:

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test, T_train, T_test, y0_train, y0_test, y1_train, y1_test =
train_test_split( x, Y, T, y0, y1,test_size=0.2, random_state=42 )
```
*Слика7: Поделба на тренинг и тест множество*

На слика 7 ја правиме таа поделба. Воведиваме нови променливи:

X_train, X_test ги чуваат карактеристиките на пациентите, се добиваат со поделба на табелата X

Y_train, Y_test се вистинските измерени резултати на пациентите, се добиваат со поделба на променливата Y

T_train, T_test се информациите за тоа дали пациентот примил третман или не

y0_train, y0_test, y1_train, y1_test се предвидени/измерени информации за секој пациент соодветно. Важно е да се напомене дека y0_train и y1_train **НЕМА НИКОГАШ ДА ГИ КОРИСТИМЕ**. За податоците од видот y0_*** и y1_*** ни требаат само тест множествата.

Откако го поделивме податочното множество можеме да започнеме со креирање на моделите. Првиот модел кој ќе го имплементираме е Causal Forest и на слика 8 може да се види како го креираме и тренираме.

```
estimator = CausalForest(n_trees=300,
model_T=DecisionTreeRegressor(),
model_Y=DecisionTreeRegressor(),
random_state=42)

estimator.fit(Y=Y_train,
T=T_train,
W=X_train,
X=X_train,
inference='blb')
```
*Слика8: Креирање и тренирање на Causal Forest модел*

За креирањето на моделот дефинираме шума од 300 дрва, каде што за моделирање на третманот (model_T) и исходот (model_Y) се користат

Decision Tree Regressor. Потоа, го трениpаме моделот со estimator.fit() каде што:

Y=Y_train ова е здравствениот исход, Causal Forest ќе учи како третманот влијае на оваа променлива

T=T_train променлива на третманот, целта е да се измери каузалниот ефект на третманот врз Y

W=X_train ковариjати за моделирање на третманот, тие се користат за да се корегира веројатноста некој да добие третман (за да не дојде до пристрасност)

X=X_train карактеристики за хетерогеност на ефектот, со нив моделот учи кај кого третманот има посилен ефект

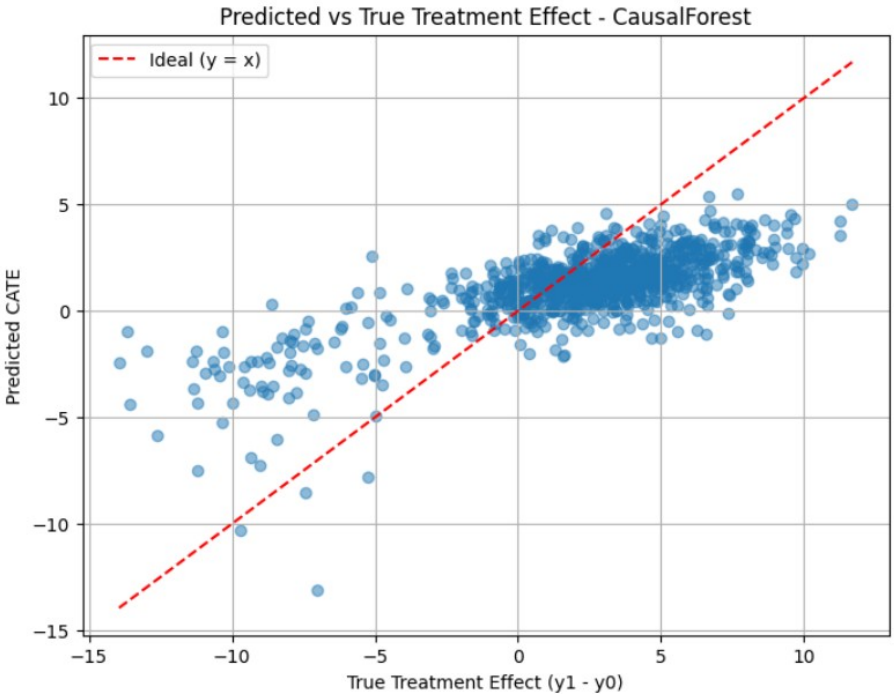
inference='blb' за статистичка инференција ќе користиме **Bag of Little Bootstraps**

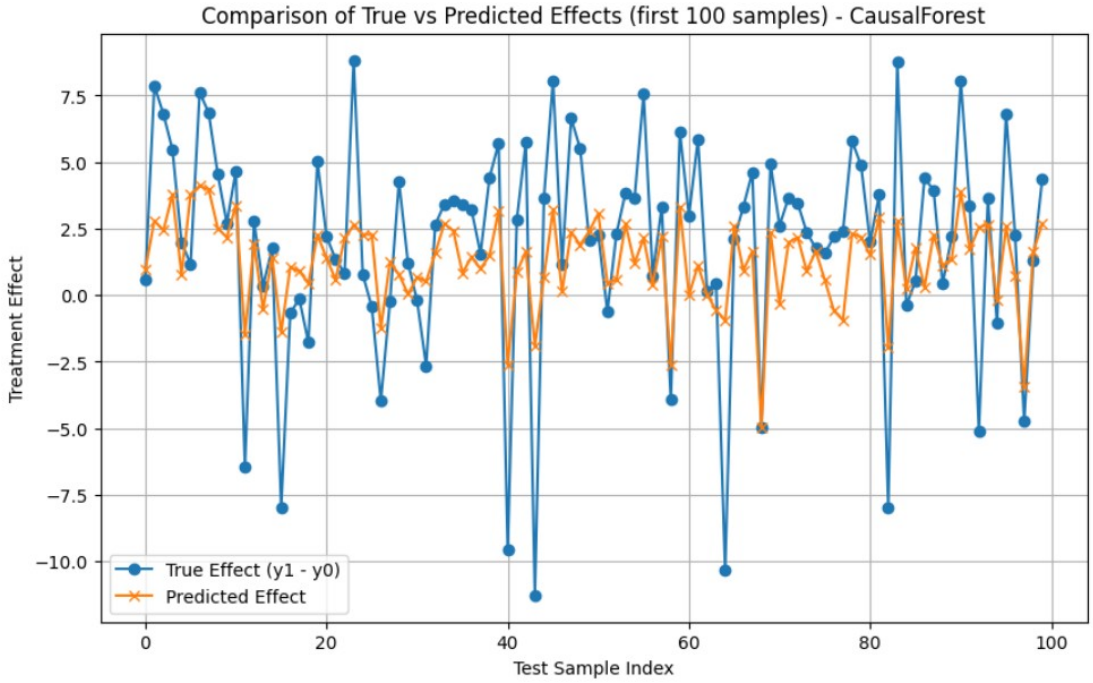
Сега кога веќе го истрениравме моделот, можеме да ја направиме евалуацијата. (слика 9) Во првиот ред од кодот ја пресметуваме вистинската каузална разлика *ground truth treatment effect* за тест сетот. Во вториот ред од кодот се пресметува предвидената каузална разлика од моделот Causal Forest и во третиот ред се пресметува просечна грешка на проценетите каузални ефекти во споредба со вистинските. Колку е помала просечната грешка толку резултатите се подобри.

```
true_effect_test = y1_test - y0_test
pred_effect_causal = estimator.effect(X_test)
rmse = sqrt(mean_squared_error(true_effect_test, pred_effect_causal))
```

*Слика9: Евалуација на Causal Forest моделот*

*Слика10: Визуелизација на точноста на Causal Forest. Црвената испрекината линија е совршена линија y = x – колку повеќе точките се блиску до неа, толку подобар е моделот.*




*Слика11: Визуелизација на грешките на моделот Causal Forest. Со сина боја можеме да го видиме точниот ефект додека пак со портокалова можеме да ги видиме грешките на Causal Forest*

Просечната грешка изнесува 3.127 што не е ниту голема ниту мала грешка. Вредностите на вистинскиот ефект (true_effect_test) се движат од -13.95 до 11, што значи дека RMSE од 3.11 претставува релативно умерена грешка. Исто така на сликите 10 и 11 може да се забележат две визуелизации за тоа колку ефектот од третманот на моделот отстапува од вистинскиот ефект.

Откако завршивме со Causal Forest продолжуваме со имплементација на Double Machine Learning. Она што е различно во споредба со Causal Forest е што овој модел има потреба од скалирање на атрибутите доколку тие се движат во различни интервали. DML не е толку робусен на нив колку што е Causal Forest. Затоа прво правиме скалирање на X_train и X_test а, потоа креирање на DML модел и негово тренирање. (слика 12)

```
est = DML(model_y=GradientBoostingRegressor(),
model_t=GradientBoostingRegressor(),
model_final=LassoCV(fit_intercept=False))

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

est.fit(Y=Y_train,
T=T_train,
W=X_train_scaled,
X=X_train_scaled)
```

*Слика12: Креирање на DML модел, скалирање на коваријатите и тренирање на моделот*

За креирање на DML моделот ги користиме следните аргументи:

model_y=GradientBoostingRegressor()
Ова е моделот што ја учи врската меѓу коваријатите X и исходот Y, користиме Gradient Boosting Regressor

model_t=GradientBoostingRegressor()
Ова е моделот што ја учи врската меѓу коваријатите X и третманот T.

Целта е да се процени колку веројатно е некој да добие третман врз основа на ковариј атите. Повторно користиме Gradient Boosting.

model_final=LassoCV(fit_intercept=False)
Модел што ја проценува каузалната врска помеѓу T и Y, откако претходните два модели ги „отстраниле“ ефектите на X.
Користиме Lasso регресија со вкрстена валидација (LassoCV).
fit_intercept=False значи дека не се додава константен член, затоа што DML веќе работи со прочистени податоци (остатоци).

За тренирање на DML моделот го користиме истиот код како за Causal Forest со тоа што сега ги земаме скалираните вредности на ковариј атите.
За овој модел ги пресметуваме и ласо коефициентите со цел да ја определиме важноста на карактеристиките и ги визуелизираме пресметките. (слика 13) Потоа, пресметуваме rmse на DML моделот и добиваме резултат од 2.1065 (слика 14)
Споредено со Causal Forest оваа грешка е многу помала што значи дека DML е многу поуспешен од него.

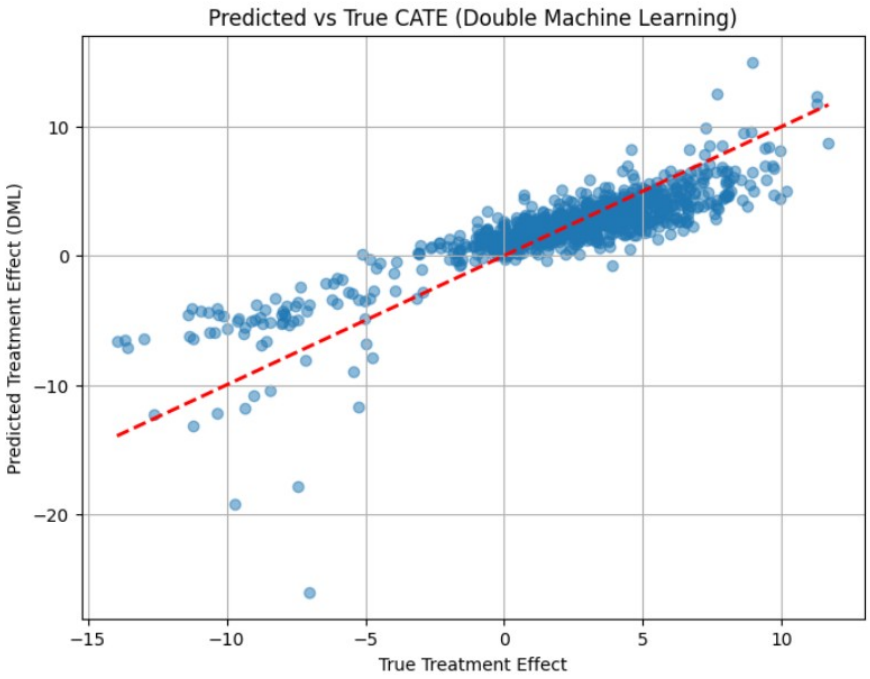
```
[ 2.02798425 -0.02766267 -0.         -0.15476103 -0.         -0.
  0.         -0.         -2.93610172  0.         -0.          0.08157005
 -0.         -0.         -0.         -0.         -0.         -0.05325
 -0.         -0.23582874  1.11620566  0.         -0.         -0.14070538
 -0.08893018 -0.          0.         -0.          0.06703431 -0.01824004
 -0.0109638   0.         -0.         -0.          0.         -0.
  0.          0.         -0.04959471 -0.51422666  0.18669315  0.91909757
 -0.         -0.         -0.          0.          0.          0.
 -0.         -0.         -0.         -0.0545401  -0.07634918  0.
 -0.01788622 -0.13985262]
```

*Слика13: Прикажување на Lasso коефициентите на DML моделот (со овие коефициенти можеме да видиме колку придонесува секој атрибут за предвидувањата на DML)*

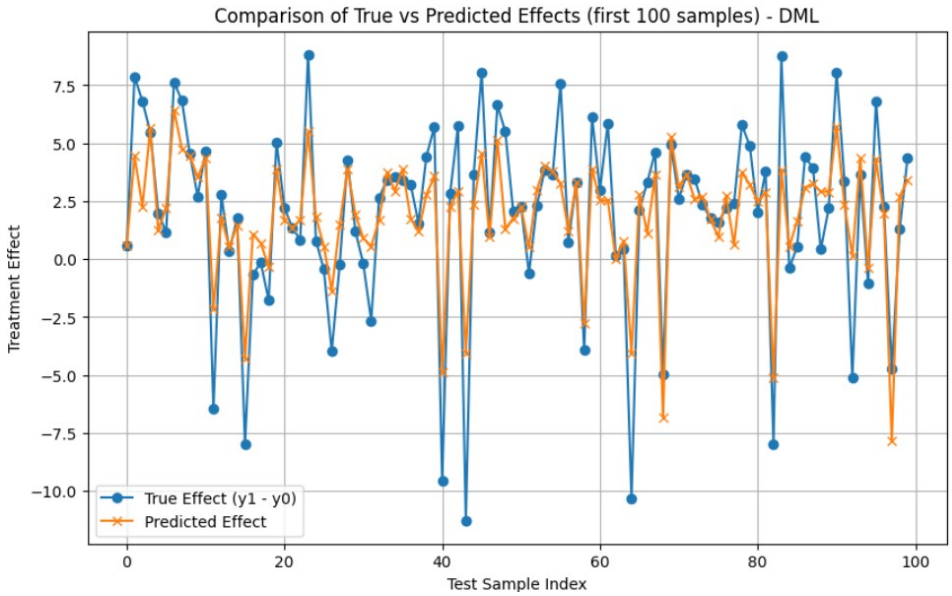
```
pred_effect_dml = est.effect(X_test_scaled)
rmse = sqrt(mean_squared_error(true_effect_test, pred_effect_dml))
```

*Слика14: Евалуација на DML*

Како и за претходниот модел така и за овој, ја визуелизираме грешката за да видиме колку отстапува од вистинските вредности (слики 15 и 16)



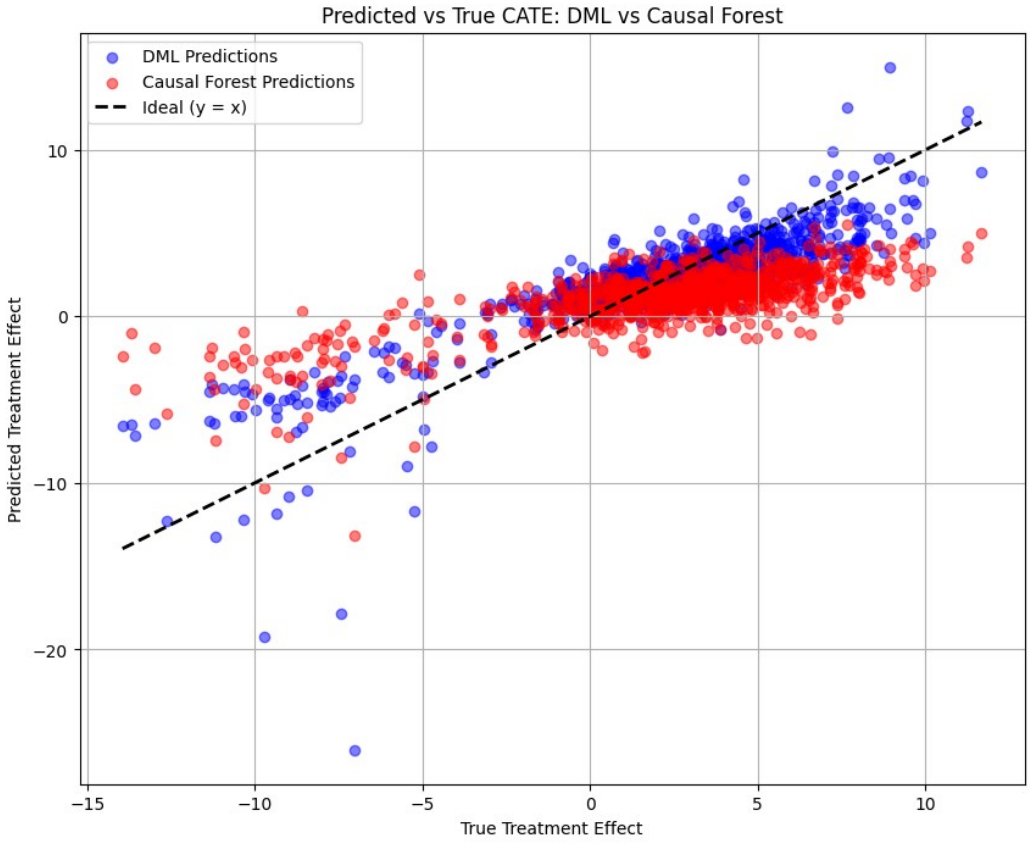

*Слика15: Визуелизација на точноста на моделот DML.*
*Црвената испрекината линија е совршена линија y = x – колку повеќе точките се блиску до неа, толку подобар е моделот.*




*Слика16: Визуелизација на грешките на моделот DML.*
*Со сина боја можеме да го видиме точниот ефект додека пак со портокалова можеме да ги видиме грешките на DML.*

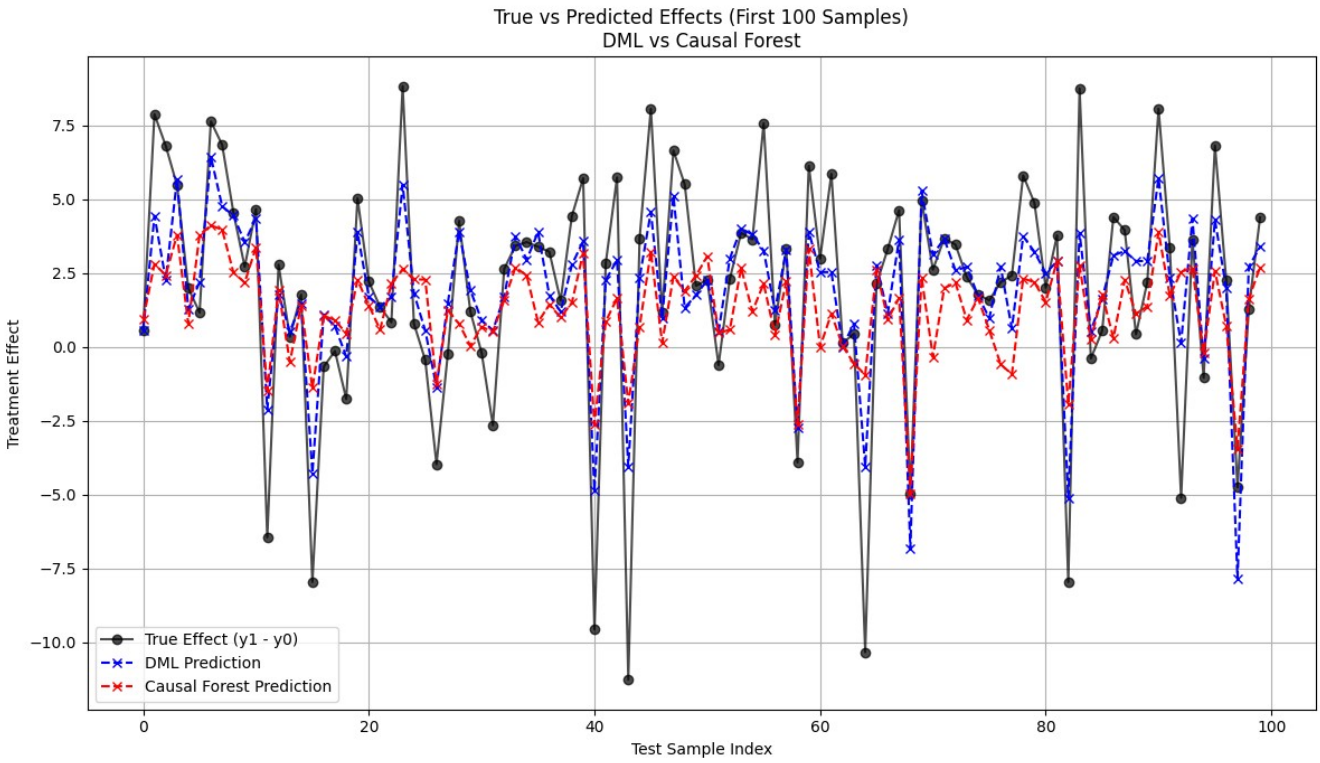
На слика 15 гледаме дека предвидувањата на DML се поблиску до совршената линија отколку предвидувањата на Causal Forest. Слично забележуваме и на слика 16. Грешката е помала во споредба на таа кај Causal Forest.

И за крај на оваа имплементација за двата модели на каузална инференција, ја илустрираме точноста на двата модели на еден график со цел полесна споредба. На слика 17 може да се зебелажи потполно истиот график кој го илустриравме погоре но овојпат и двата модели се на ист цртеж.




*Слика17: Предвидувањата на DML vs предвидувањата на Causal Forest. Точките на DML (сините точки) се поблиску до совршената линија во споредба со точките на Causal Forest (црвените точки)*

А, на слика 18 може да се забележи визуелизација на различностите на првите 100 примероци од предвидениот ефект на Causal Forest и DML со точниот ефект од третманот.




*Слика18: Споредба на точниот ефект на каузалната инференција со предвидениот ефект на Causal Forest и DML (на првите 100 примероци од тест множеството)*

# ЗАКЛУЧОК

Заклучокот од експериментот покажува дека иако и двата модели – Causal Forest и Double Machine Learning (DML) – успешно се користат за проценка на каузални ефекти, во овој случај DML дава подобри резултати. Според метриката RMSE, DML постигна пониска грешка (2.1065) во споредба со Causal Forest (3.1274), што укажува дека DML попрецизно ги проценува индивидуалните каузални ефекти. Ова може да се должи на комбинацијата на флексибилни модели за Y и T (Gradient Boosting) и стабилната финална регресија (LassoCV), кои подобро ја искористија структурата на податоците. Затоа, во дадениот сет на податоци, DML се покажува како поадекватен пристап за проценка на CATE.

## ПРИМЕНА

Каузалната инференција наоѓа широка примена во различни индустрии и сектори. Во медицината, се користи за да се процени ефектот на нови терапии или лекови кај различни групи пациенти. Во економијата и финансиите, се применува за анализа на влијанието на фискални политики, субвенции или каматни стапки врз економските резултати. Во маркетингот, овие методи помагаат да се открие како различни стратегии (како промоции или реклами) влијаат врз однесувањето на потрошувачите. Исто така, каузалната инференција е значајна во јавните политики, каде што се користи за мерење на ефектот од образовни програми, социјални интервенции и регулативи врз општеството.

# РЕФЕРЕНЦИ


Causal Forest Information

Causal Inference Information

Double Machine Learning Information

Dataset Github