Київський національний університет імені Тараса Шевченка

Факультет комп’ютерних наук та кібернетики

Кафедра математичної інформатики

Звіт з теми:

“Прогнозування розміру доходів”

Виконали групи МІ-4 Омельяненко Андрій, Стецик Олексій

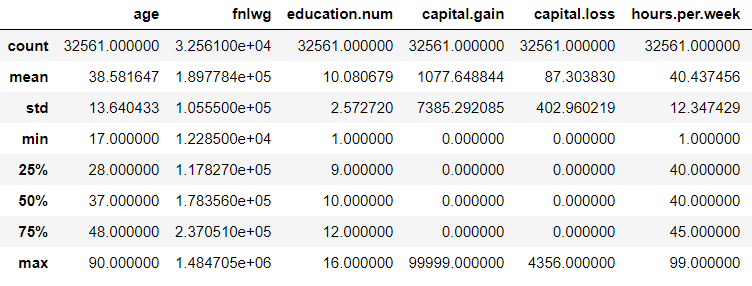
**Аналіз та формалізація задачі**

Задача полягає в аналізі та класифікації заробітної плати дорослих в США, використовуючи UCI Machine Learning Repository Adult data set(<http://archive.ics.uci.edu/ml/datasets/Adult>).

В нашому розпорядженні є дані про 32560 людей. Кожна людина характеризується 14 атрибутами(8 номінальних та 6 числових) та бінарним значенням target(0 – людина отримує менше 50000 $/year, 1 -не менше).

Числові атрибути: вік, фінальний коефіцієнт(~кількість людей з такими характеристиками в країні), найвищий рівень освіти(відсортований), отриманий капітал, втрачений капітал, кількість робочих годин на тиждень.

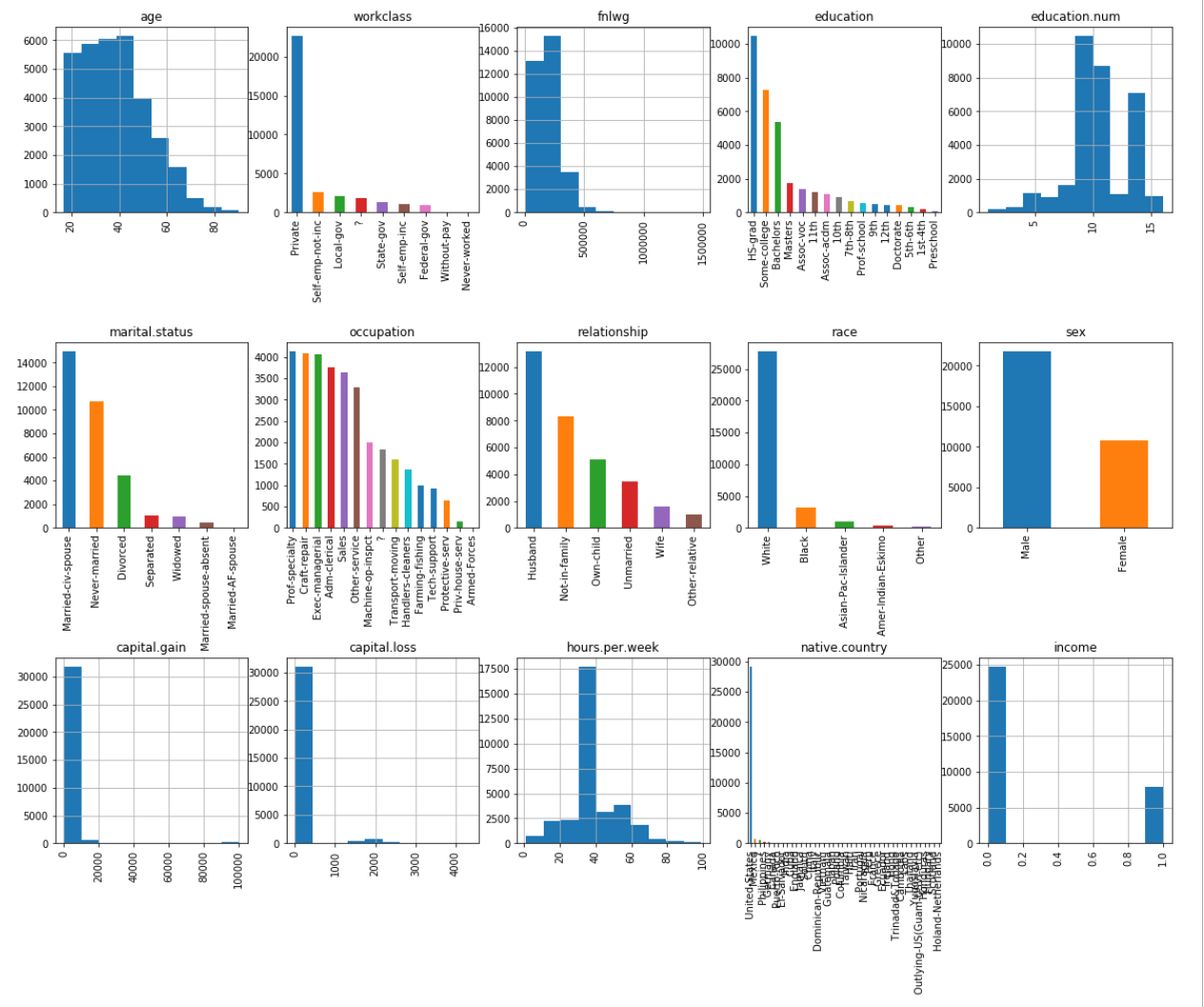
Статистика по числових атрибутах:

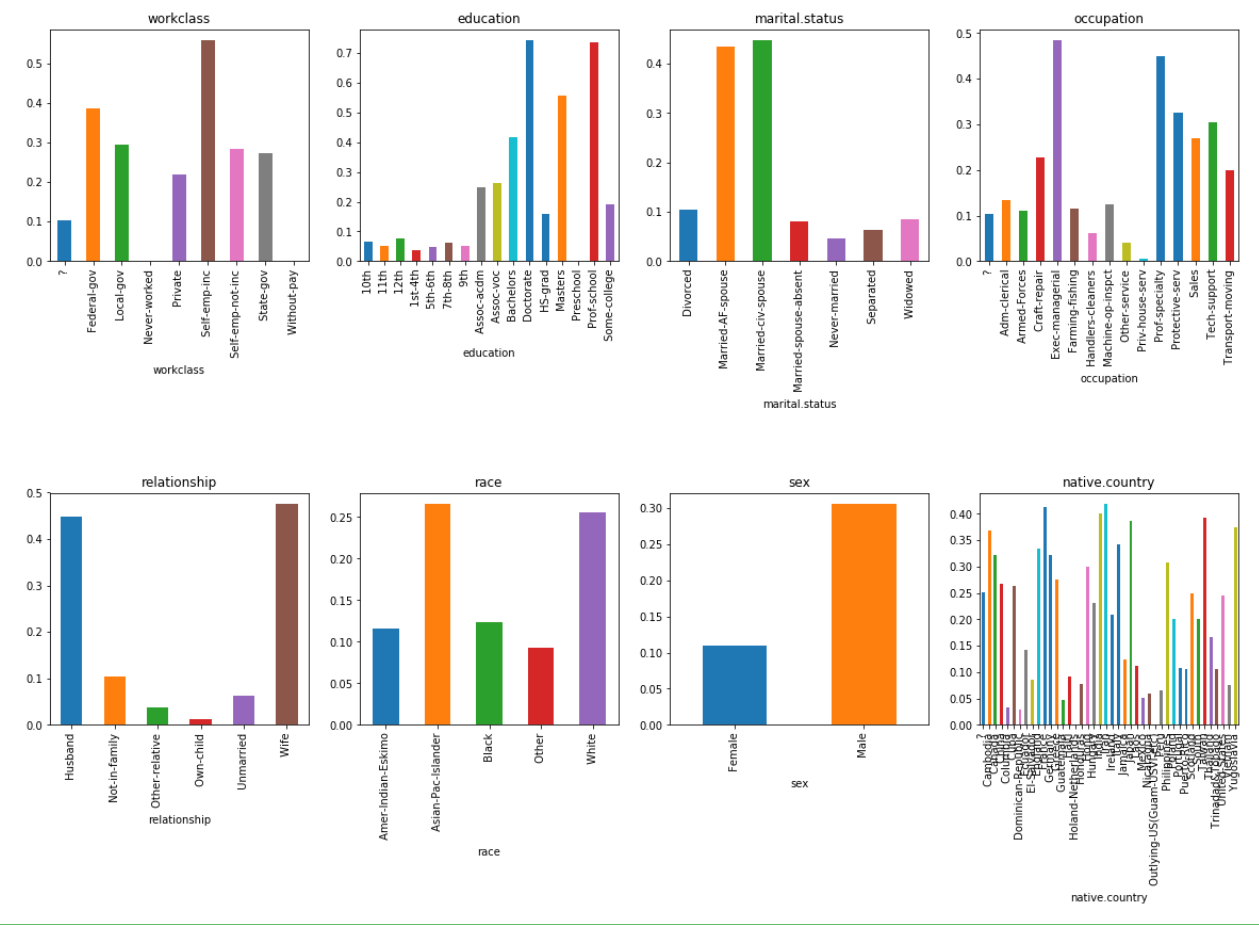


Номінальні атрибути: робочий клас(описує тип роботи(самостійний, державний, приватний і тд), найвищий рівень освіти, сімейний статус(одружений, неодружений і тд), вид діяльності(сільське господарство, менеджмент і тд), відносини(неодружений, дружина і тд), раса, стать, батьківщина.

Статистика по номінальних атрибутах:

* Освіта: 32% закінчили школу, 22% були студентами коледжу і 16.5% мають ступень Бакалавра.
* Сімейний стан: 46% одружені, 33% ніколи не були одруженими, 14% розлучені.
* Відносини: 40% одружені чоловіки, 26% не в сім’ї.
* Раса: 86% білі, 10% чорні.
* Стать: 67% чоловіки, 33% жінки.
* Батьківщина: 90% США, 2% невідомо

Розглянемо розподіл кожної функції, щоб ми могли краще розуміти, що відбувається в наших даних. Зображаємо кількість рядків кожного типу для кожного атрибуту.

Також розглянемо який відсоток людей для кожного значення кожного атрибута має >50000$/year(0.3 ~ 30%). 

Наступним кроком розглянемо кореляцію між різними атрибутами. Рахувати кореляцію між номінальними атрибутами неможливо, тому ми їх закодуємо. Це робиться з-за допомогою LabelEncoder з sickit-learn. Для кращого результату ми для кожного атрибуту сортуємо можливі значення в порядку зростання в залежності від % людей що мають більше 50000$/year.

Приклад до workclass, аналогічно робиться для всіх атрибутів:

workclass

Never-worked 0.000000

Without-pay 0.000000

? 0.104031

Private 0.218673

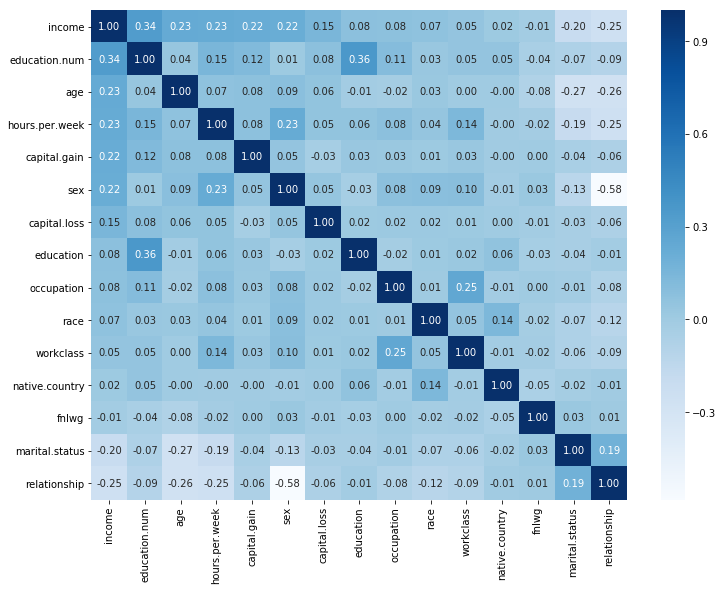
State-gov 0.271957

Self-emp-not-inc 0.284927

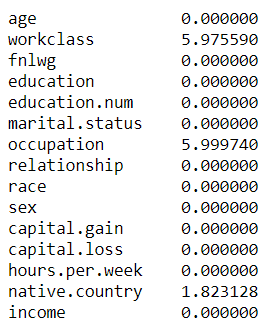
Local-gov 0.294792

Federal-gov 0.386458

Self-emp-inc 0.557348

Матриця кореляції між кожною парою атрибутів. 

**Попередня обробка даних**

В першу чергу перевіримо якісь наших даних: 

Отже, є 3 атрибути з невідомими/відсутніми значеннями. Робочий клас(6%), батьківщина(2%) та вид діяльності(6%). Робочий клас в 70% випадків є приватним сектором, тому можна для кожного з цих 6% приписати приватний сектор. Аналогічно, батьківщиною в 90% випадків є США, тому можна приписати США для всіх невідомих значень. Оскільки ми маємо недостатньо даних для людей батьківщина яких не США, має зміст розділити ці дані на 2: USA та not-USA або ж взагалі видалити цей атрибут.

Отже, для 2х атрибутів невідомі значення замінились медіанним значенням на вибірці. На жаль, для виду діяльності не можна провести підчистку аналогічним чином, оскільки цей атрибут не має переважаючої характеристики та може бути дуже важливим. Оскільки вхідних невідомих даних цього атрибуту всього 6% то їх можна просто видалити.

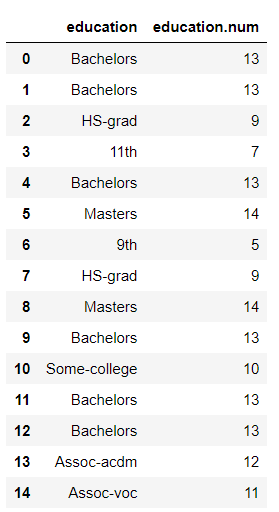
Крім цього дані містять 47 рядків-дублікатів, їх можна просто видалити.

На жаль, існує дисбаланс у пропорції атрибуту-таргету, маємо 75% <= 50K та 25% >50K, але цього достатньо для побудови достатньо точних класифікаторів.

Для використання моделей машинного навчання необхідно щоб кожен атрибут був числовим, тому переведемо *income* та *sex* в бінарні числа наступним чином:

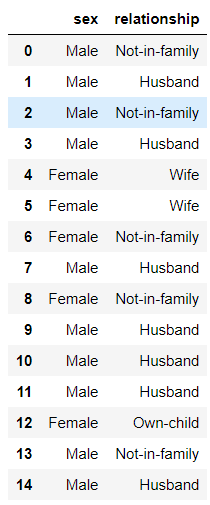
* data['income'] = data['income'].map({' <=50K': 0, ' >50K': 1}).astype(int)#(Якщо <=50000$/year то 0, інакше 1)
* data['sex'] = data['sex'].map({' Female': 0, ' Male': 1}).astype(int)#(Якщо жінка то 0, інакше 1)

Можна побачити велику додатну кореляцію між education та education.num, дивимось що означають ці стовпці:

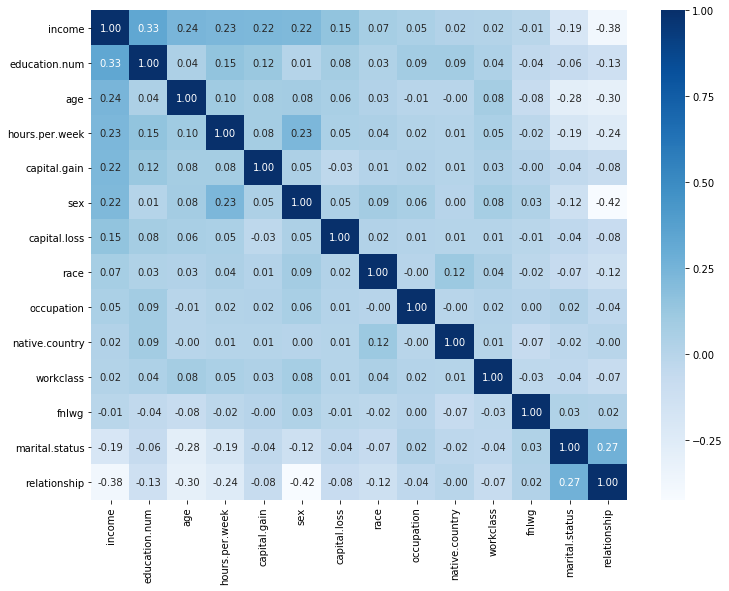


По суті це одне й теж, але в education.num ми маємо властивість чисел між собою: більше число – більший освітній рівень, тому видалимо рядкове позначення, а числове залишимо, адже їхня властивість може бути корисною для моделей.

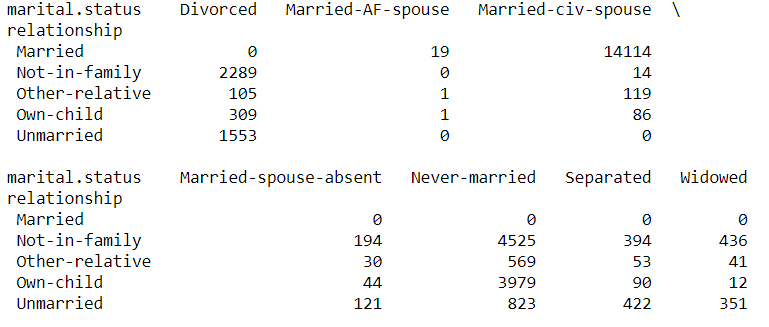
Побачимо також велику від’ємну кореляцію між відносинами та статтю, поглянемо чому так:



Зрозуміло що ця кореляція з’являється між Male-Husband та Female-Wife, тому виконаємо наступну підміну: Husband, Wife -> Married. Таблиця кореляції змінилась: кореляція між *sex* та *relationship* зменшилась.

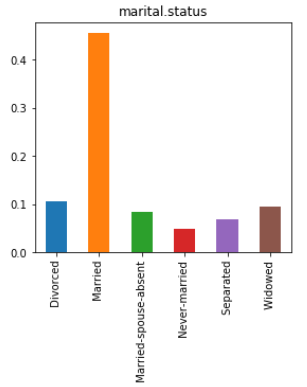
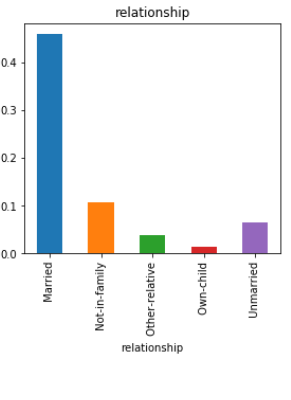


Але кореляція між *relationship* та *marital.status* збільшилась, подивимось чому:



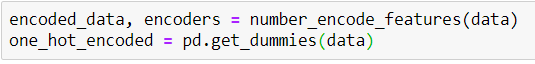
Бачимо що Married-AF-spouse майже завжди Married, тому просто заміняємо Married-AF-Spouse, Married-civ-spouse на Married.

Існують люди які не одружені в одному атрибуті, але одружені по-іншому атрибуту. Цим і іншими подібними неточностями можна знехтувати, оскільки це присутнє в невеликій кількості рядків і не буде критичним атрибутом. Можна спробувати залишити лише однин атрибут з *marital.status* та *relationship*, та розділити на Married, not-Married, оскільки середній відсоток людей в цих вибірках що отримують >50K приблизно однаковий.

Також має сенс об’єднани ці 2 атрибути в один, що матиме поля Married та not-Married.

Наступне що потрібно зробити – перевести всі номінальні дані в числові. Це можна зробити з-за допомогою LabelEncoder. Також існує альтернативний підхід – one-hot-encoding, перевід всіх номінальних атрибутів в бінарні(pd.get\_dummies).



В подальшому ми будемо використовувати лише encoded\_data, оскільки one\_hot\_encoded показує гірші результати.

Атрибут fnlwg – означає вагу даного рядка(~ скільки людей в США підпадають під таку характеристику). Для нашої задачі класифікації цей атрибут не дає корисної інформації, тому видаляємо його.

Наші дані розбиваються на навчальну та тестову вибірку в відношенні 70:30.

**Огляд можливих методів машинного навчання для даної задачі.**

З допомогою пакету sklern ми порівняли результати роботи різних алгоритмів класифіукації.

Детальний аналіз дати був проведений з допомогою jupyter notebook.

Посилання на сам notebook: <https://github.com/Omelianenko/adult_salary_prediction/blob/master/main_adult.ipynb>

Моделі, які були досліджені:

*LogisticRegression*. Логістична регресія не враховує зв’язок між різними атрибутами, тому її використання дає результат гірше, ніж інші моделі.

*Баєсівський класифікатор*. Також дає погану точність, оскільки так само, як і логістична регресія, використовує припущення про незалежність атрибутів.

*NeuralNetwork*. Нейронні мережі враховують зв’язок між атрибутами, що не враховувалось ні в логістичній регресії, ні в баєсівському класифікаторі, тому результат отримали краще. Після експерементів з різною кількістю шарів, прийшли до висновку, що при 3 прихованих шарах результат роботи найкращий. При збільшенні кількості шарів результат покращується повільно.

*SVM*. В загальному має працювати з нелінійними даними. Оскільки ми маємо нелінійні атрибути, такі як capial gain/loss, тому зробили висновок про доцільність використання SVM, але на практиці результат далеко не найкращий.

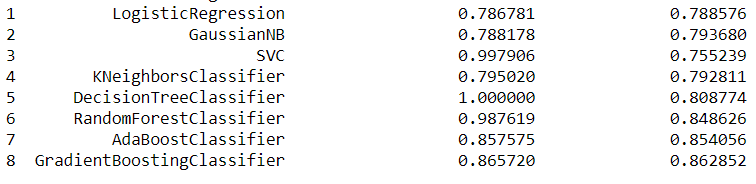
*KNN*. Використовуючи KNN класифікатор ми серйозно покращили точність, але необхідно було погратись з значенням K. При K = 23 – найкращий результат для тестових даних.

*DecisionTree*. Дозволяє отримувати точність 1.0 на тренувальних даних, але завжди присутній оверфіт, можна покращувати результат даного класифікатора з-за допомогою інших алгоритмів підрізання.

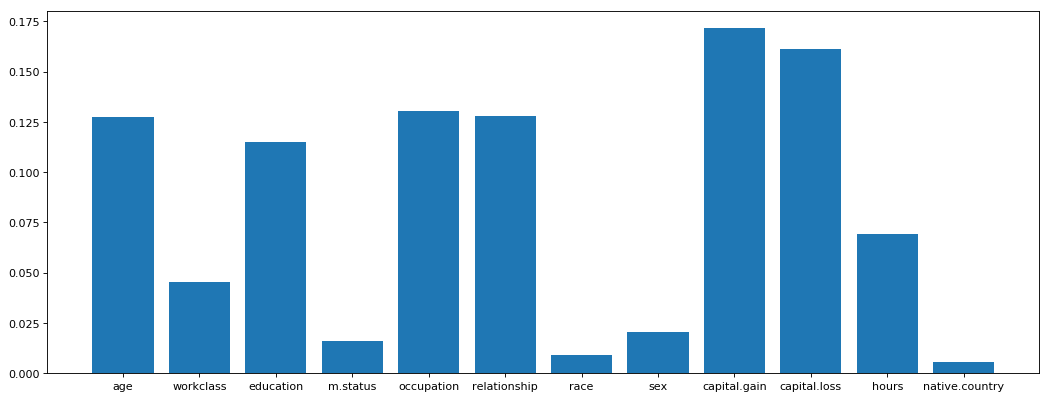
*RandomForest*. Як і очікувалось, перевершив *DecisionTree.*

*GradientBoostingClassifier*. Дав найкращий результат, також допоміг краще зрозуміти, які атрибути мають найбільший вплив на класифікацію, як виявилось це наступні характеристики: *married, capital gain, hours per week* і *age*.

Статистика для різних моделей:



Вплив атрибутів від GradientBoostingClassifier



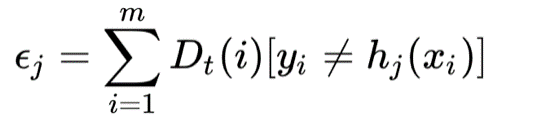
Ми зосередилися на двох методах на адабусті і на нейроній мережі. Адабуст це алгоритм машинного навчання який може використовуватися з декількома алгоритмами класифікації для того щоб покращувати результат. Адабуст є адаптивним класифікатором в тому розумінні що на кожному новому кроці будується класифікатор з більшим коефіцієнтом на тих даних, які були невірно розпізнанними на попередньому кроці. Порядок роботи алгоритму адабуст такий: він викликає слабкі класифікатори в циклі. Після кожного виклику оновлюється розприділення ваг які відповідають за важливість об’єктів тієї множини яка навчається.

Запишемо формально даний алгоритм:

Дано: 

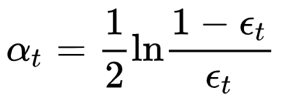
Початкова ініціалізація: 

Ітируючися по циклу знаходимо найменшу зважену помилку:

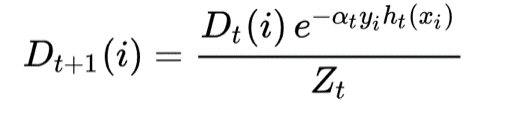


Якщо ця помилка >= 0.5 то зупиняємося.

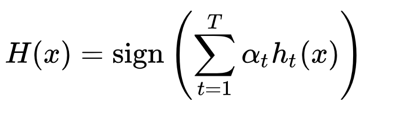
Беремо таке значення:



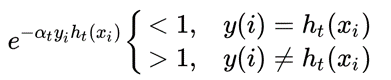
Оновлюємо ваги:



І власне отримуємо результуючий класифікатор



Умова яка буде виконуватися, тому і адабуст буде працювати.

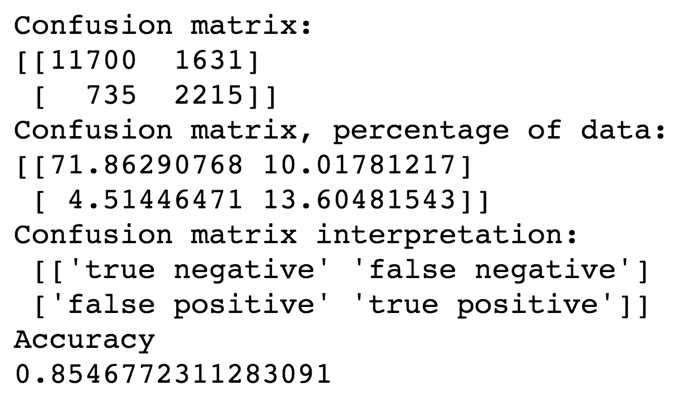


Чому вибрали саме адабуст, тому що він дуже добре будує композиції, а по даних було видно, що залежність композиційна. Також його просто реалізовувати.

Тепер про нейрону мережу.

Ми використовували keras для її реалізації.

А саме функцією активації був гіперболічний тангенс, функцією втрати була квадратична похибка, а оптимізатор був ададельта. Кількість прихованих шарів, які найкраще відповідають даній задачі була 3. Чому ми вибрали саме ададельту, цей алгоритм фактично є покращенням sgd(stochastic gradient boosting), а саме коли ми бачимо що градієн йде в одному напрямі, то дистанція на яку він буде переходити буде значно збільшуватися. Це дає значну оптимізацію над sgd. Tanh вибрали тому що sigmoid міг би застопоритися. Коли на вхід сигмоїду дають строго негативні значення він видає значення дуже близькі до нуля, в цьому проблема.



Як ми бачимо даний метод працює краще ніж адабуст, проте довше і навчається. Ми використовували 50 епох для навчання і crossvalidation set розміром 10 процентів.

Таким чином ми бачимо що найкраще працюють ті алгоритми які найбільше пристосовані до композиційних даних. Коли ми починаємо підкручувати гіпотезу, то точність падає. Звідси робимо висновок що сильно більше з даних моделей отримати ми не можемо.

**Висновок**. Ми залишились задоволеними нашим результатом, найкращий результат отримав GradientBoostingClassifier – 86.1% успіху. Хоч класифікатор SVM I KNN і круто відпрацьовують на навчальних вибірках, на тестових вони показують гірший результат, хоча для покращення SVM можна погратись з коефіцієнтом лямда(послабити штраф за неточність для тренувальної вибірки). Найкращі результати було отримано наступними моделями: нейронна мережа з 3 прихованими шарами, адабуст та GradientBoosting. Подальші удосконалення можна робити підкруючи параметри різних класифікаторів та переведенням певних номінальних атрибутів в бінарні значення(one-hot-encoding).