УДК 004.85
DOI: 10.18523/2617-3808.2019.2.32-37

Глибовець А. М., Мухопад О. О.

ВИКОРИСТАННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ СТВОРЕННЯ АНАЛІТИЧНОЇ ПЛАТФОРМИ НЕРУХОМОСТІ В УКРАЇНІ

У цій роботі розглянуто процес побудови моделі машинного навчання для аналізу вартості нерухомості, який передбачає пошук та підготовку вхідних даних, вибір, тренування моделі машинного навчання та її оптимізацію за допомогою ApacheSpark.

Ключові слова: аналіз даних, машинне навчання, gradient boosting trees, GBT, Spark.

Вступ

Останнім часом методи машинного навчання використовують для розв'язку різноманітних задач. Однією з галузей, де сучасні підходи впроваджуються в реальне життя, є ринок нерухомості. Найвідоміші ріелторські та девелоперські компанії мають штати аналітиків даних, які дають їм змогу оптимізувати бізнес-процеси, обирати найбільш вигідні місця для початку будівництва, орієнтуватися на цільові аудиторії та багато іншого. Тому в цій роботі ми розглянемо процес розробки моделі машинного навчання для передбачення ціни нерухомості оренди квартир у Києві з використанням регресійних алгоритмів.

Створення тренувального набору

Для того щоб займатися аналітикою, спочатку необхідно отримати достатній за об'ємом набір тренувальних даних. На жаль, у відкритому доступі готових датасетів нерухомості України не було знайдено, тому було вирішено побудувати датасет самостійно.

Портал оголошень нерухомості «dom.ria» надає доступ до API [3], за допомогою якого і було вирішено будувати датасет. Для роботу з API було обрано мову Python 3.7 через відносну простоту та мінімум необхідної інфраструктури для виконання запитів.

Приклад датасету, що був отриманий:

```
id,area,rooms,floor,total_floors,wall_type,district,city,state,metro,metro_branch,furnished,
heating,repaired,balcony,jacuzzi,view,price
14965129,45,2,3,9,панель,Святошинский,Киев,Киевская,Святошин,3,1,1,0,0,0,0,8000.0
11959304,130,3,16,25,кирпич,Днепровский,Киев,Киевская,Левобережная,3,1,1,1,0,0,1,68722.5
14199425,132,3,6,9,кирпич,Шевченковский,Киев,Киевская,,,1,1,1,0,0,0,35750.0
13438557,61,1,24,25,монолітно-кирпичный,Святошинский,Киев,Киевская,Житомирская,3,1,0,1,0,0,0,12000.0
8379840,57,2,23,32,монолитно-кирпичный,Дарницкий,Киев,Киевская,,,1,1,1,0,0,0,12500.0
15232311,156,3,4,12,кирпич,Голосеевский,Киев,Киевская,Площадь Льва Толстого,2,1,1,1,0,1,1,55000.0
15459801,55,2,18,23,кирпич,Печерский,Киев,Киевская,Печерская,1,1,1,1,0,0,0,33000.0
15458345,150,4,7,7,кирпич,Оболонский,Киев,Киевская,,,1,1,1,0,1,0,44000.0
```

Загалом вийшло 7012 унікальних записів.

Підготовка даних

Для проведення успішних експериментів важливим кроком є відбір характеристик і підготовка даних. Спочатку було прибрано колонки «місто» і «область», оскільки наш датасет містить тільки квартири Києва.

Позаяк колонки «з мебеллю», «з опаленням», «з ремонтом», «з балконом», «з джакузі» необов'язкові, вони містили багато пропущених даних. Ці колонки також були видалені з тренувальної вибірки.

Також дані мають пряму залежність між гілкою/станцією метро та між площею/кількістю кімнат (рис. 1). Це не дає нашій моделі нової інформації, тому ці ознаки також були прибрані.

Найкориснішою для нас виявилася колонка «загальна площа» (рис. 2). Ми бачимо пряму залежність площі від ціни. І навпаки, поверх, на якому розташована квартира, та кількість поверхів у будинку лише дотично вказують нам на інші фактори, як-от панельний це будинок чи «сталінка» (рис. 3). Також ми можемо простежити закономірність між колонкою «тип стін» та розміром орендної плати (рис. 4).

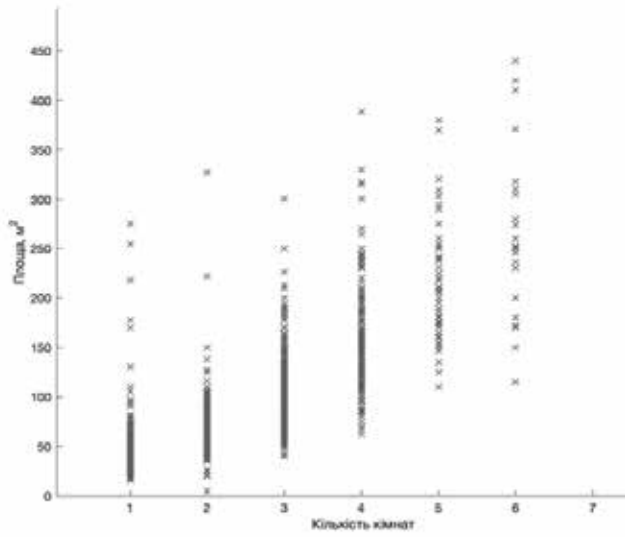

Рис. 1. Залежність площі від кількості кімнат

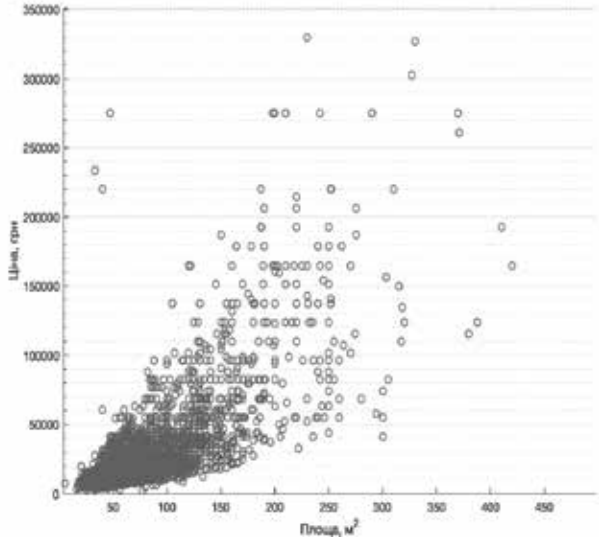

Рис. 2. Залежність ціни оренди від загальної площі квартири

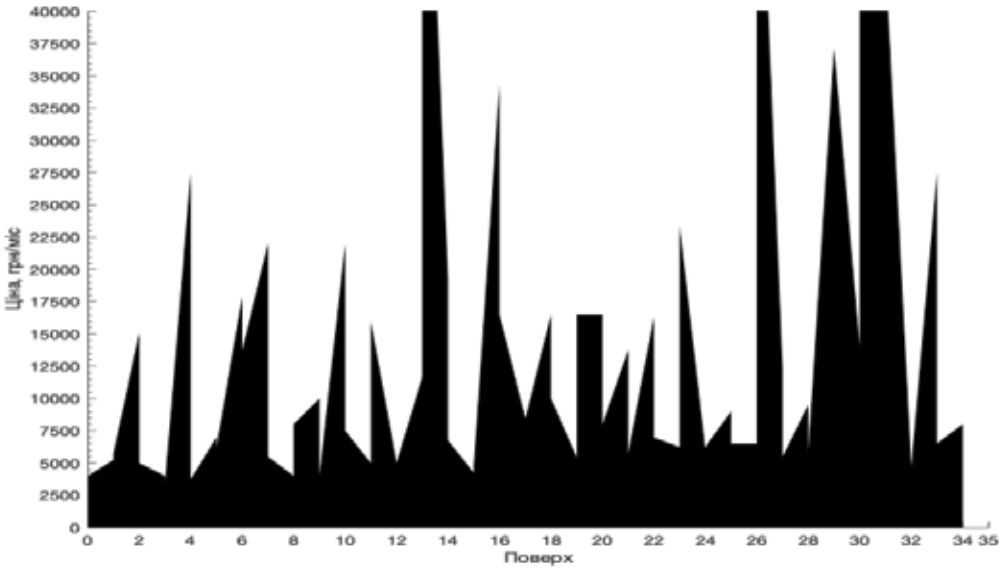

Рис. 3. Залежність ціни оренди від поверху будинку

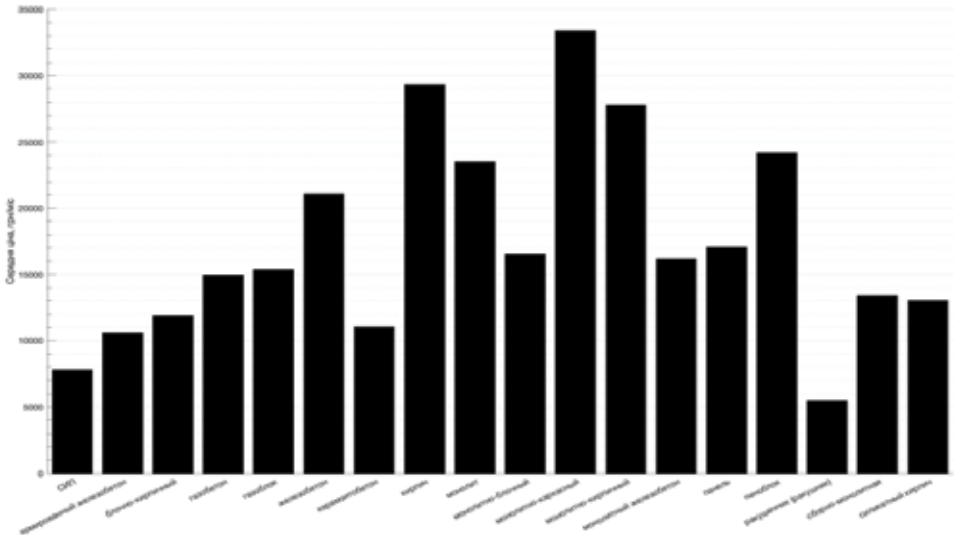

Рис. 4. Залежність середньої ціни оренди від типу стін будинку

Завантажимо датасет і відберемо необхідні колонки. Для опрацювання даних будемо використовувати ApacheSpark [1] з мовою Scala.

```
valdf = spark.read.format("csv")
    .option("header", "true")
    .option("inferSchema", "true")
    .load("apartments_kyiv.csv")
valdataset = df
    .withColumn("labels", 'price.cast("Double"))
    .select('district, 'area, 'metro, 'wall_type, 'labels)
    .where('metro.isNotNull&& 'district.isNotNull)
    .cache()
```

Ми завантажуємо датасет із параметром inferSchema для автоматичного визначення типів. Далі, робимо явне перетворення колонки у тип double як новий стовпчик та ігноруємо рядки з незаповненим районом і метро.

Підготовлений датасет має такий вигляд:

```
dataset.show(10, truncate = false)
+------------+-----+--------------------+------------------+-------+
|district    |area |metro               |wall_type         |labels |
+------------+-----+--------------------+------------------+-------+
|Святошинский|45.0 |Святошин            |панель            |8000.0 |
|Днепровский |130.0|Левобережная        |кирпич            |68722.5|
|Святошинский|61.0 |Житомирская         |монолитно-кирпичный|12000.0|
|Голосеевский|156.0|Площадь Льва Толстого|кирпич            |55000.0|
|Печерский   |55.0 |Печерская           |кирпич            |33000.0|
|Голосеевский|47.0 |Выставочный центр   |кирпич            |14000.0|
|Дарницкий   |43.0 |Осокорки            |панель            |9000.0 |
|Голосеевский|110.0|Дворец Украина      |кирпич            |30250.0|
|Днепровский |92.0 |Дарница             |кирпич            |20000.0|
|Голосеевский|65.0 |Демиевская          |кирпич            |17000.0|
+------------+-----+--------------------+------------------+-------+
```

Для того, щоб використати датасет у моделі машинного навчання, нам потрібно перетворити його у дві колонки: числовий вектор атрибутів і значення ціни. Але у нашому датасеті поля district, metro та wall_type є стрічками. Для перетворення цих колонок ми використали StringIndexer бібліотеки SparkMLlib та OneHotEncoder [2].

На виході ми отримали набір масштабованих параметрів від 0 до 1.

```
+---------------------------------+-----------------------------+
|assembled_features               |features                     |
+---------------------------------+-----------------------------+
|(78,[0,7,16,61],[75.0,1.0,1.0,1.0])|[0.16666666666666666,0.0...] |
|(78,[0,6,30,61],[32.0,1.0,1.0,1.0])|[0.04519774011299435,0.0...] |
|(78,[0,1,31,61],[45.0,1.0,1.0,1.0])|[0.08192090395480225,1.0...] |
|(78,[0,9,14,61],[68.0,1.0,1.0,1.0])|[0.14689265536723164,0.0...] |
|(78,[0,8,26,61],[65.0,1.0,1.0,1.0])|[0.1384180790960452,0.0...]  |
|(78,[0,4,13,62],[36.0,1.0,1.0,1.0])|[0.05649717514124294,0.0...] |
|(78,[0,3,11,62],[37.0,1.0,1.0,1.0])|[0.059322033898305086,0.0...]|
|(78,[0,1,31,61],[47.0,1.0,1.0,1.0])|[0.08757062146892655,1.0...] |
+---------------------------------+-----------------------------+
```

Вектор features готовий бути використаним в алгоритмах машинного навчання.

Вибір алгоритму машинного навчання

Найпростіший алгоритмом, що може бути використаний для розв'язку нашої задачі, є лінійна регресія. Переваги: простий для розуміння алгоритм; порівняно швидкий. Недоліки: велика кількість даних потребує дослідження взаємозв'язку вхідних атрибутів; допускає негативні значення, що для нашого продукту недопустимо.

RandomForest: базується на деревах прийняття рішень. Переваги: збільшують зміщення (bias) системи, що допомагає позбавитися перетренування; за великої кількості атрибутів для кожного дерева використовує різні набори атрибутів; відносно легко налаштувати. Недоліки: зменшує дисперсію (variance) системи, що може нашкодити недотренованим системам.

GradientBoostingTrees: також базуються на деревах прийняття рішень, проте дають можливість знаходити розв'язок на незбалансованих наборах даних шляхом посилення впливу необхідної змінної. Переваги: теоретично підходять для будь-якої задачі; оптимізація під час тренування. Недоліки: схильні до перетренування; важкі в налаштуванні; працюють довше, ніж RandomForest.

Виходячи із природи вхідних даних та поставленої задачі, ми зупинилися на GradientBoostingTrees.

Проведення експериментів

Для проведення експериментів створимо екземпляр класу GBTRegressor:

```
valregressor = newGBTRegressor()
    .setFeaturesCol("features")
    .setLabelCol("labels")
    .setMaxDepth(30)
    .setMaxIter(10)
```

Ми визначили кількість ітерацій 10 та максимальну глибину дерева 30. Глибоке дерево збільшує дисперсію системи.

Для того щоб правильно навчити модель, потрібно обрати метрику оцінки якості роботи нашої моделі. Для навчання моделі було обрано крос-валідацію з середньоквадратичною помилкою RMSE [4].

SparkMLlib дає змогу проводити крос-валідацію не тільки вхідних даних, а й параметрів налаштування алгоритму. Цей підхід називають GridSearch. Для початку, ми створюємо сітку параметрів ParamGrid. У ньому зазначаємо різну інтенсивність градієнтного спуску та мінімальну кількість інформації, яку надає кожен атрибут, щоб можна було розділити дерево.

Створимо екземпляр класу для крос-валідації. Ми вказуємо наш пайплайн із перетворенням датасету та алгоритмом навчання, під'єднуємо метрики для оцінки моделей та передаємо сам датасет.

```
valmodel = newCrossValidator()
   .setEstimatorParamMaps(paramGrid)
    .setEstimator(pipeline)
    .setEvaluator(evaluator)
    .setNumFolds(7)
    .fit(dataset)
```

Після процесу крос-валідації зміна model містить найкращу модель, керуючись оцінкою evaluator, тобто з ряду моделей із різними варіантами параметрів ми отримуємо на виході модель з найменшою середньоквадратичною похибкою (RMSE).

Передбачення ціни та оцінка якості моделі

Дістанемо параметри найкращої моделі після крос-валідації:

```
valmetrics=model.getEstimatorParamMaps
  .zip(model.avgMetrics)
  .maxBy(_._2)._1
```

Результат:

```
gbtr_b4b0b6a658ff-minInfoGain: 0.1,
gbtr_b4b0b6a658ff-stepSize: 1.0
```

Як ми бачимо, у найкращої моделі крок градієнтного спуску дорівнює максимальному значенню 1, а оптимальний мінімум інформації для розділу дерева – 0.1.

Маючи готову натреновану модель, ми можемо робити передбачення:

```
model.transform(testData)
```

де testData - Dataset із тестовими даними для передбачення.

Самі дані мають бути необробленими, тобто містити таку саму інформацію. Всі інші перетворення пайплайн бере на себе. Зразок передбачення:

```
+--------------------+-------+------------------+
|            features| labels|        prediction|
+--------------------+-------+------------------+
|[0.06779661016949...| 6999.0|          9264.875|
|[0.09039548022598...|35750.0|           35750.0|
|[0.11299435028248...| 9000.0|            9000.0|
|[0.11016949152542...| 7500.0| 7499.722463269417|
|[0.11299435028248...|24750.0|20687.222463269416|
|[0.03389830508474...| 6500.0|  6715.97856801726|
|[0.18644067796610...|15000.0|14999.722463269414|
|[0.14406779661016...|10000.0| 9999.888519140672|
+--------------------+-------+------------------+
```

Для оцінки якості передбачення використаємо evaluator, який ми визначили раніше. Також нам знадобиться середнє арифметичне істинної ціни.

```
valrmse = evaluator.evaluate(testData)
valmean = testData.select(avg('labels)).first().
getDouble(0)
```

Ефективною метрикою якості передбачення є коефіцієнт варіації (Coefficientofvariation, CV). Якщо RMSE показує абсолютну помилку в масштабі передбачуваної величини, то CV – відносну. Коефіцієнт варіації розраховують за формулою:

$$CV = \frac{RMSE}{mean} \times 100.$$

Для нашого передбачення ми отримали такі оцінки:

$RMSE$ = 1646.4734 грн
$mean$ = 14437.375 грн
CV = 11.404 %.

Значення CV задовільне. Іншими словами, за вартості квартири в 10 тис. грн, система може помилятись всього на +/– 1140 грн.

Незважаючи на недосконаліть вхідних параметрів для аналізу та відсутність головних ціноутворювальних факторів, таких, як «відстань до метро», «стан квартири» або суб'єктивних параметрів, як-от «якість ремонту», «дизайн» тощо, GradientBoostingTree разом з методикою крос-валідації змогли добитися непоганих результатів.

Висновки

У цій роботі ми розглянули повний процес побудови моделі машинного навчання на прикладі аналітики нерухомості м. Києва, який передбачає пошук і підготовку вхідних даних, вибір, тренування моделі машинного навчання та її оцінку, використовуючи ApacheSpark.

Головною проблемою є те, що дані, які ми здобули, не були високої якості. Маючи багато необов'язкових полів, ці дані також не містили головних ціноутворювальних факторів.

Як шлях для покращення проекту і подальшої розробки продукту в цілому, ми плануємо обрахувати за допомогою наявних API та додати до тренувального датасету відстань до найближчого метро та/або відстань до центру міста міським транспортом. Ця інформація знизила би рівень похибки передбачення.

Проте загалом ми маємо на виході робочу модель, яка досить непогано передбачає ціну нерухомості.

Список літератури

1. ApacheSpark 2.4.0 Documentation [Electronic resource]. – Mode of access: https://spark.apache.org/. – Title from the screen.
2. Extracting, transformingandselecting features [Electronic resource]. – Mode of access: https://spark.apache.org/docs/latest/ml-features.html. – Title from the screen.
3. Ria.com для розробників [Electronic resource]. – Mode of access: https://developers.ria.com/. – Title from the screen.
4. Root-mean-square deviation. [Electronic resource]. – Mode of access: https://en.wikipedia.org/wiki/Root-mean-square_deviation. – Title from the screen.

References

ApacheSpark 2.4.0 Documentation. Retrieved from https://spark.apache.org/.

Extracting, transforming and selecting features. Retrieved from https://spark.apache.org/docs/latest/ml-features.html.

Ria.com for developers. Retrieved from https://developers.ria.com/.

Root-mean-squaredeviation. Retrieved from https://en.wikipedia.org/wiki/Root-mean-square_deviation.

A. Hlybovets, O. Mukhopad

MACHINE LEARNING TECHNIQUES TO CREATE AN ANALYTICAL REAL ESTATE PLATFORM IN UKRAINE

Recently, methods of machine learning are used to solve various problems. One of the areas where modern approaches are being introduced in real life is the real estate market. The most reputable real estate and development companies have state-of-the-art data analysts that allow them to optimize business processes, choose the most advantageous places to start construction, target their audience, and much more. Therefore, this paper considers the process of developing a model of machine learning to predict the price of real estate rental apartments in Kyiv by using regression algorithms.

We describe the process of constructing a model of machine learning for the analysis of the value of real estate, which includes the search and preparation of a dataset, features choice, training of the machine learning model and its optimization with the help of Apache Spark, an open-source framefork for distributed computing.

In order to engage in analytics, it is first necessary to obtain a sufficient amount of the training data. Unfortunately, we do not find it in open access to the ready-made datasets of real estate. We decided to build a dataset.

For successful experiments, an important step is to select characteristics and prepare data. At first, the columns "city" and "area" are removed, as our dataset contains only apartments in Kyiv.

Since the columns "with furniture", "with heating", "for repair", "with a balcony", "with a jacuzzi" are optional, they contained many missing data. These columns are also removed from the training sample.

In addition, the data have a direct relationship between the branch / subway station and the area / number of rooms. This does not give our model a new information, so these attributes are removed.

We analyzed such ML algorithms: linear regression, Random Forest, Gradient Boosting Trees. Based on the nature of the training dataset and our goals, we decided to use Gradient Boosting Trees.

The "dom.ria" API was used to build the dataset. The data has been cleared and normalized. After that, the Gradient Boosting Trees model is trained by using techniques called grid search and cross validation. The Coefficient of variation was selected for the prediction of the quality metric. It received a value of 11,404 %.

Keywords: machine learning, real estate, Gradient Boosting Trees, Apache Spark.

Матеріал надійшов 22.04.2019