Практическая работа по Прикладному Искусственному Интеллекту №2

Прогнозирование цен на жилье с помощью нейросетевой регрессионной модели

ℹ️ **Выполнил:**
студент группы R32362 Шишков Кирилл Алексеевич

Преподаватель прикладного искусственного интеллекта:
Евстафьев Олег Александрович

Университет ИТМО, Факультет Систем Управления и Робототехники
Санкт-Петербург, 2022

Цель работы

По имеющимся данным о ценах на жильё предсказать окончательную цену каждого дома с учетом характеристик домов с использованием нейронной сети.

Описание выполненной работы

Для выполнения работы был написан следующий код:

Datalore App: Collaborative Data Science Platform by JetBrains

Use smart coding assistance for Python, SQL, R and Scala in Jupyter notebooks, run code on powerful CPUs and GPUs, collaborate with your team, and easily share the results.

https://datalore.jetbrains.com/notebook/Sp94OFFf9Z4bjSufliaBWf/jTAwS99rEI17POMj4lkSqX/

Результаты выполнения работы

Результат MSE, Adam(0.01), epochs = 40, batch_size = 10

	Id	SalePrice
0	1461	152169.8
1	1462	172104.12
2	1463	188756.61
3	1464	188918.47
4	1465	164018.17
5	1466	168853.8
6	1467	170104.27
7	1468	148978.5
8	1469	178426.03

Сравнение

```
epoch = 1, batch_size = 10, Adam = 0.001, score(loss, mae) = [8259936768.0, 50176.51171875]
epoch = 1, batch_size = 10, Adam = 0.01, score(loss, mae) = [3119443456.0, 37112.1484375]
epoch = 1, batch_size = 10, Adam = 0.1, score(loss, mae) = [8800959488.0, 77004.140625]
epoch = 1, batch_size = 10, Adam = 0.5, score(loss, mae) = [10346195968.0, 83019.8671875]
epoch = 10, batch_size = 10, Adam = 0.001, score(loss, mae) = [3239927808.0, 31291.759765625]
epoch = 10, batch_size = 10, Adam = 0.01, score(loss, mae) = [4947524096.0, 48606.921875]
epoch = 10, batch_size = 10, Adam = 0.1, score(loss, mae) = [3401975808.0, 33618.3828125]
epoch = 10, batch_size = 10, Adam = 0.5, score(loss, mae) = [3916904960.0, 40449.9921875]
epoch = 20, batch_size = 10, Adam = 0.001, score(loss, mae) = [3768697600.0, 31525.5]
epoch = 20, batch_size = 10, Adam = 0.01, score(loss, mae) = [4222941952.0, 29894.8046875]
epoch = 20, batch_size = 10, Adam = 0.1, score(loss, mae) = [3823480576.0, 34994.5234375]
epoch = 20, batch_size = 10, Adam = 0.5, score(loss, mae) = [21608966144.0, 126324.3046875]
epoch = 40, batch_size = 10, Adam = 0.001, score(loss, mae) = [3858196224.0, 30329.83203125]
epoch = 40, batch_size = 10, Adam = 0.01, score(loss, mae) = [4222112000.0, 29041.61328125]
epoch = 40, batch_size = 10, Adam = 0.1, score(loss, mae) = [3752135168.0, 30532.798828125]
epoch = 40, batch_size = 10, Adam = 0.5, score(loss, mae) = [5670837760.0, 55540.4296875]
epoch = 1, batch_size = 50, Adam = 0.001, score(loss, mae) = [34336749568.0, 170268.828125]
epoch = 1, batch_size = 50, Adam = 0.01, score(loss, mae) = [9474880512.0, 67340.15625]
epoch = 1, batch_size = 50, Adam = 0.1, score(loss, mae) = [3239941888.0, 39564.55859375]
epoch = 1, batch_size = 50, Adam = 0.5, score(loss, mae) = [9717193728.0, 81851.359375]
epoch = 10, batch_size = 50, Adam = 0.001, score(loss, mae) = [4381291520.0, 42929.4609375]
epoch = 10, batch_size = 50, Adam = 0.01, score(loss, mae) = [3875172352.0, 31601.94140625]
epoch = 10, batch_size = 50, Adam = 0.1, score(loss, mae) = [5328695296.0, 52897.1796875]
epoch = 10, batch_size = 50, Adam = 0.5, score(loss, mae) = [4625627136.0, 44500.62109375]
epoch = 20, batch_size = 50, Adam = 0.001, score(loss, mae) = [3102276864.0, 33122.53125]
epoch = 20, batch_size = 50, Adam = 0.01, score(loss, mae) = [4168937984.0, 33866.12890625]
epoch = 20, batch_size = 50, Adam = 0.1, score(loss, mae) = [4997595136.0, 47671.078125]
epoch = 20, batch_size = 50, Adam = 0.5, score(loss, mae) = [6242327552.0, 60588.42578125]
epoch = 40, batch_size = 50, Adam = 0.001, score(loss, mae) = [3751676928.0, 31271.326171875]
epoch = 40, batch_size = 50, Adam = 0.01, score(loss, mae) = [4334851584.0, 29796.923828125]
epoch = 40, batch_size = 50, Adam = 0.1, score(loss, mae) = [3801958400.0, 28690.3984375]
epoch = 40, batch_size = 50, Adam = 0.5, score(loss, mae) = [4023978752.0, 33321.5546875]
epoch = 1, batch_size = 100, Adam = 0.001, score(loss, mae) = [37190647808.0, 178028.15625]
epoch = 1, batch_size = 100, Adam = 0.01, score(loss, mae) = [10349047808.0, 50743.97265625]
epoch = 1, batch_size = 100, Adam = 0.1, score(loss, mae) = [14047434752.0, 97786.1328125]
epoch = 1, batch_size = 100, Adam = 0.5, score(loss, mae) = [24734474240.0, 140290.703125]
epoch = 10, batch_size = 100, Adam = 0.001, score(loss, mae) = [6935681024.0, 46532.47265625]
epoch = 10, batch_size = 100, Adam = 0.01, score(loss, mae) = [3916981760.0, 32839.34765625]
epoch = 10, batch_size = 100, Adam = 0.1, score(loss, mae) = [3868989952.0, 32963.94921875]
```

```
epoch = 10, batch_size = 100, Adam = 0.5, score(loss, mae) = [3800810240.0, 35847.1875]
epoch = 20, batch_size = 100, Adam = 0.001, score(loss, mae) = [3951766528.0, 42090.8984375]
epoch = 20, batch_size = 100, Adam = 0.01, score(loss, mae) = [4190109184.0, 31602.6328125]
epoch = 20, batch_size = 100, Adam = 0.1, score(loss, mae) = [4198629376.0, 37522.984375]
epoch = 20, batch_size = 100, Adam = 0.5, score(loss, mae) = [4109541376.0, 32343.595703125]
epoch = 40, batch_size = 100, Adam = 0.001, score(loss, mae) = [3165615616.0, 32179.572265625]
epoch = 40, batch_size = 100, Adam = 0.01, score(loss, mae) = [4399861760.0, 29514.03125]
epoch = 40, batch_size = 100, Adam = 0.1, score(loss, mae) = [4520254464.0, 32013.404296875]
epoch = 40, batch_size = 100, Adam = 0.5, score(loss, mae) = [3937315328.0, 31291.671875]
epoch = 1, batch_size = 200, Adam = 0.001, score(loss, mae) = [38275551232.0, 180835.125]
epoch = 1, batch_size = 200, Adam = 0.01, score(loss, mae) = [23606384640.0, 137013.296875]
epoch = 1, batch_size = 200, Adam = 0.1, score(loss, mae) = [34924134400.0, 171919.484375]
epoch = 1, batch_size = 200, Adam = 0.5, score(loss, mae) = [480475414528.0, 629365.0]
epoch = 10, batch_size = 200, Adam = 0.001, score(loss, mae) = [14690016256.0, 100197.90625]
epoch = 10, batch_size = 200, Adam = 0.01, score(loss, mae) = [3223928832.0, 38320.4296875]
epoch = 10, batch_size = 200, Adam = 0.1, score(loss, mae) = [3671956992.0, 34034.74609375]
epoch = 10, batch_size = 200, Adam = 0.5, score(loss, mae) = [39209644032.0, 183180.953125]
epoch = 20, batch_size = 200, Adam = 0.001, score(loss, mae) = [6726608384.0, 46277.04296875]
epoch = 20, batch_size = 200, Adam = 0.01, score(loss, mae) = [3754701312.0, 32777.703125]
epoch = 20, batch_size = 200, Adam = 0.1, score(loss, mae) = [3983741440.0, 32925.546875]
epoch = 20, batch_size = 200, Adam = 0.5, score(loss, mae) = [39206412288.0, 183172.140625]
epoch = 40, batch_size = 200, Adam = 0.001, score(loss, mae) = [3709298944.0, 41114.20703125]
epoch = 40, batch_size = 200, Adam = 0.01, score(loss, mae) = [4491483648.0, 30676.0078125]
epoch = 40, batch_size = 200, Adam = 0.1, score(loss, mae) = [4319334400.0, 30454.158203125]
epoch = 40, batch_size = 200, Adam = 0.5, score(loss, mae) = [39197470720.0, 183147.734375]
```

Заключение

Самое точное предсказание цены на квартиру получилось при количестве эпох 18, размере батча 10 и значении оптимизатора Адам 0.1. Средняя точность получилась равна примерно 15%.

Ответы на вопросы

Как выше перечисленные параметры влияют на полученный вами результат?

Что такое эпоха (Epoch)? В чем отличие от итерации (Iteration)?

Эпоха - время обучения нейронной сети, за которое полностью прогоняется весь набор входных данных. Итерация, в свою очередь, количество батчей, необходимое для завершения одной эпохи. Батч - одна из групп, на которые делятся эпохи.

Что такое функция активации? Какие вам известны? Как и зачем используются в нейронной сети?

Функция активации - функция, которая реализует выходной блок нейросети.

`relu` - Спрямленный линейный блок.

При значениях по умолчанию он возвращает по элементам. max(x, 0). В противном случае: f(x) = max_value для x >= max_value, f(x) = x для threshold <= x < max_value, f(x) = alpha * (x — threshold).

Аргументы

- **x:** Входной тензор.
- **alpha:** float. Наклон отрицательной части. По умолчанию ноль.
- **max_value:** float. Порог насыщения.
- **threshold:** float. Пороговое значение для пороговой активации.

Возвращает

Тензор.

`selu` - Масштабируемая экспоненциальная линейная единица.

SELU равна: scale * elu(x, alpha), где apha и scale являются предопределенными константами. Значения alpha и scale выбираются таким образом, чтобы среднее и дисперсия входов сохранялись между двумя последовательными слоями до тех пор, пока веса правильно инициализированы, а количество входов «large enough».

Аргументы

- **x:** Тензор или переменная для расчета функции активации.

Возвращает

Масштабированную экспоненциальную активацию: scale * elu(x, alpha).

Функция активации `softsign`.

Аргументы

- **x:** Входной тензор.

Возвращает

Активация softsign: x / (abs(x) + 1).

Функция активации `Softplus`.

Аргументы

- **x:** Входной тензор.

Возвращает

активация Softplus: log(exp(x) + 1).

`elu` - Экспоненциальный линейный блок.

Аргументы

- **x:** Входной тензор.
- **alpha:** Скаляр, наклон отрицательного сечения.

Возвращает

Экспоненциальную линейную активацию: x если x > 0 и alpha * (exp(x)-1) если x < 0.

Функция активации `Softmax`.

Аргументы

- **x:** Входной тензор.
- **axis:** Integer, ось, по которой применяется нормализация софтмакса.

Возвращает

Тензор, выход софтмакс-трансформации.

Что такое MSE(Mean Squared Error) - Средняя квадратичная ошибка? Что такое MAE(Mean Absolute Error)? Для чего используются.

`MAE`**(Mean Absolute Error) -** Метрика измеряет среднюю сумму абсолютной разницы между фактическим значением и прогнозируемым значением.

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|e_t|, where \quad e_t = original_t - predict_t$$

`MSE` **(Mean Squared Error) -** Метрика измеряет среднюю сумму квадратной разности между фактическим значением и прогнозируемым значением для всех точек данных. Выполняется возведение во вторую степень, поэтому отрицательные значения не компенсируют положительными. А также в силу свойств этой метрики, усиливается влияние ошибок, по квадратуре от исходного значения. Это значит, что если в в исходных измерениях мы ошиблись на 1, то метрика покажет 1, 2-4, 3-9 и так далее. Чем меньше MSE, тем точнее наше предсказание. Оптимум достигается в точке 0, то есть мы идеально предсказываем.

$$MSE = \frac{1}{n}\sum_{t=1}^{n} e_t^2, where \quad e_t = original_t - predict_t$$