Линейная регрессия: прогноз оклада по описанию вакансии

Данное задание основано на материалах лекций по линейной регрессии и посвящено предсказанию оклада, исходя из описания вакансии.

Вы научитесь:

- использовать линейную регрессию
- применять линейную регрессию к текстовым данным

Введение

Линейные методы хорошо подходят для работы с разреженными данными — к таковым относятся, например, тексты. Это можно объяснить высокой скоростью обучения и небольшим количеством параметров, благодаря чему удается избежать переобучения.

Линейная регрессия имеет несколько разновидностей в зависимости от того, какой регуляризатор используется. Мы будем работать с гребневой регрессией, где применяется квадратичный, или L2-регуляризатор.

Реализация в Scikit-Learn

Для извлечения TF-IDF-признаков из текстов воспользуйтесь классом sklearn.feature_extraction.text.TfidfVectorizer.

Для предсказания целевой переменной мы будем использовать гребневую регрессию, которая реализована в классе sklearn.linear_model.Ridge.

Обратите внимание, что признаки LocationNormalized и ContractTime являются строковыми, и поэтому с ними нельзя работать напрямую. Такие нечисловые признаки с неупорядоченными значениями называют категориальными или номинальными. Типичный подход к их обработке — кодирование категориального признака с m возможными значениями с помощью m бинарных признаков. Каждый бинарный признак соответствует одному из возможных значений категориального признака и является индикатором того, что на данном объекте он принимает данное значение. Данный подход иногда называют one-hot-кодированием. Воспользуйтесь им, чтобы перекодировать признаки LocationNormalized и ContractTime. Он уже реализован в классе sklearn.feature_extraction.DictVectorizer. Пример использования:

```
from sklearn.feature_extraction import DictVectorizer
enc = DictVectorizer()
X_train_categ = enc.fit_transform(
    data_train[['LocationNormalized', 'ContractTime']]
        .to_dict('records'))
X_test_categ = enc.transform(
    data_test[['LocationNormalized', 'ContractTime']]
        .to_dict('records'))
```

Вам понадобится производить замену пропущенных значений на специальные строковые величины (например, 'nan'). Для этого подходит следующий код:

```
data_train['LocationNormalized'].fillna('nan', inplace=True)
data_train['ContractTime'].fillna('nan', inplace=True)
```

Инструкция по выполнению

1. Загрузите данные об описаниях вакансий и соответствующих годовых зарплатах из файла salary-train.csv.
2. Проведите предобработку:
   - Приведите тексты к нижнему регистру.
   - Замените все, кроме букв и цифр, на пробелы — это облегчит дальнейшее разделение текста на слова. Для такой замены в строке text подходит следующий вызов:

```
re.sub('[^a–zA–Z0–9]', ' ', text.lower())
```

- Примените TfidfVectorizer для преобразования текстов в векторы признаков. Оставьте только те слова, которые встречаются хотя бы в 5 объектах (параметр min_df у TfidfVectorizer).
- Замените пропуски в столбцах LocationNormalized и ContractTime на специальную строку 'nan'. Код для этого был приведен выше.
- Примените DictVectorizer для получения one-hot-кодирования признаков LocationNormalized и ContractTime.
- Объедините все полученные признаки в одну матрицу "объекты-признаки". Обратите внимание, что матрицы для текстов и категориальных признаков являются разреженными. Для объединения их столбцов нужно воспользоваться функцией scipy.sparse.hstack.

3. Обучите гребневую регрессию с параметром alpha=1. Целевая переменная записана в столбце SalaryNormalized.

4. Постройте прогнозы для двух примеров из файла salary-test-mini.csv. Значения полученных прогнозов являются ответом на задание. Укажите их через пробел.

Если ответом является нецелое число, то целую и дробную часть необходимо разграничивать точкой, например, 0.42. При необходимости округляйте дробную часть до двух знаков.

Ответ на каждое задание — текстовый файл, содержащий ответ в первой строчке. Обратите внимание, что отправляемые файлы не должны содержать пустую строку в конце.