Работа № 3.2 Линейная регрессия

Цель работы: получение практических навыков построения и использования регрессионных моделей на языке Python с использованием библиотек Scikit-Learn и StatsModels.

Задание: используя программу Jupiter Notebook, язык программирования Python, библиотеки Scikit-Learn, StatsModels, NumPy, Matplotlib и др. выполнить следующие задания:

- **Парная регрессия**: построить две реализации парной линейной регрессионной модели на базе двух библиотек Scikit-Learn, StatsModels, сравнить и интерпретировать полученные результаты, входные данные рассчитать согласно варианту в таблице.
- **Множественная регрессия**: для своего варианта провести регрессионное моделирование (построить множественную регрессионную модель, ссылка для скачки данных на странице в разделе Data tables, выбрать не менее 50 строк):
  - выбрать выходную прогнозируемую переменную,
  - построить регрессионную модель со значимыми параметрами (оценить параметры межфакторной корреляции, последовательно добавлять факторы и сравнивать качество получаемых моделей, подобрать вид функции (визуальный анализ), оценить адекватность модели по статистическим показателям, каждый из этапов прокомментировать в отчете),
  - интерпретируете результаты моделирования (что значит полученная формула, какие переменные вносят больший вклад, что будет при изменении независимых переменных с зависимой),
  - прогнозировать новые значения с помощью построенной модели.

1. Теоретические сведения

Регрессия и классификация являются задачами машинного обучения с учителем. Обе используют сходную концепцию использования известных наборов данных, при этом используется алгоритм для изучения функции отображения входной переменной (x) в выходную переменную, то есть y=f(x). У задач есть общие черты и различия (Таблица).

Таблица – Сопоставление задач регрессии и классификации

	Регрессия	**Классификация**
Вид обучения	Обучение с учителем	
Задача	максимально точно аппроксимировать функцию отображения (f), чтобы при появлении новых входных данных (x) можно было прогнозировать выходные данные (y)	
Выходное значение	*числовые* или *непрерывные* (действительные числа)	*дискретные* или *категориальные*
Примеры алгоритмов	линейная регрессия, регрессия в алгоритмах опорных векторов SVR (support vector regression) и регрессионные деревья	логистическая регрессия, наивный байесовский алгоритм, решающие деревья и k-NN (k ближайших соседей)

Линейная регрессия (Linear regression) — один из самых фундаментальных алгоритмов, используемых для моделирования отношений между зависимой переменной и несколькими независимыми переменными. Целью обучения является поиск линии наилучшего соответствия.

Например: $\hat{y} = b_0 + b_1 x$ – уравнение линейной регрессии, которое описывает прямую, наиболее точно показывающую взаимосвязь между входными переменными **`x`** и выходными переменными **`y`**. Для составления этого уравнения нужно найти коэффициенты **`b`** для входных переменных.

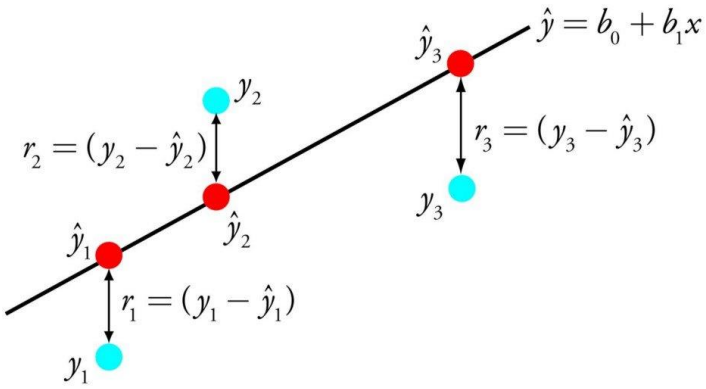

Рис. Графическое представление метода наименьших квадратов

В процессе обучения линейной регрессии находится минимизация квадратов расстояний между точками и линией наилучшего соответствия. Этот процесс известен как минимизация суммы квадратов остатков. Остаток равен разности между предсказанным значением и реальным.

$$J = \min_{b_i} \sum_{i=1}^{n} (y_i - \hat{y_i}))^2 = \min_{b_i} \sum_{i=1}^{n} (y_i - (b_0 + b_1 x_i))^2$$

Для оценки регрессионной модели используются различные методы вроде линейной алгебры или метода наименьших квадратов.

2. Задача

Рассмотрим задачу прогнозирования цен на рынке недвижимости.

- Какими данными о недвижимости мы можем располагать?
- Какие признаки влияют на цену?

Характеристики объектов недвижимости

1. Объективные характеристики:
   - *технический паспорт*
2. Субъективные характеристики (Как измерить?):
   - *состояние объекта недвижимости;*
   - *престижность района;*
   - *…*

Набор данных kc_house_data

https://www.kaggle.com/harlfoxem/housesalesprediction

Рассмотрим задачу прогнозирование цен на примере набора данных kc_house_data.

kc_house_data содержит данные о продажах индивидуальных домов в период с мая 2014 года по май 2015 в округе Кинг, штат Вашингтон, США.

Название признака	Описание
id	уникальный идентификационный номер проданного дома
date	дата продажи дома
bedrooms	количество спален
bathrooms	количество ванных комнат (где 0.25 обозначает, что комната с туалетом, 0.5 – комната с туалетом и раковиной)
sqft_living	общая площадь дома

sqft_lot	площадь прилегающей территории
floors	количество этажей
waterfront	бинарный атрибут, указывающий на то, есть ли вид на реку или нет
view	оценка внешнего вида дома (от 0 до 4)
condition	оценка состояния дома (от 0 до 5)
grade	оценка качества строительства и дизайна здания (от 1 до 13)
sqft_above	общая площадь наземной части дома
sqft_basement	общая площадь подземного части дома
yr_built	год строительства дома
yr_renovated	год последнего ремонта или последней реконструкции
zipcode	почтовый индекс дома
lat	широта
long	долгота
sqft_lot15	средняя общая площадь 15 ближайших домов
sqft_lot15	средняя площадь прилегающей территории 15 ближайших домов

3. Предобработка данных

Возможно ли уменьшить количество признаков?

Атрибуты *id*, *date, zipcode, lat, long.*

Удаляем *id*, *date, zipcode, lat, long* и sqft_basement (sqft_basement = sqft_living - sqft_above).

Пропуски в данных? Нет

Выбросы?

Формат данных: csv файл с разделителем в виде запятой. Используем библиотеку pandas.

```
import pandas as pd
from sklearn.model_selection import train_test_split

data = pd.read_csv("kc_house_data.csv", parse_dates= ['date']) # load the
data into a pandas dataframe
print(data.shape)
data.drop(['id', 'date','sqft_basement','lat','long'], axis = 1, inplace=
True)
print(data.shape)
for column in data:
print(column,end=';')
print()
# Формирование обучающей и тестовой выборок
train_data, test_data = train_test_split(data, train_size= 0.8,
random_state= 60)
train_data.astype(float).to_csv('kc_house_train_data.csv',sep=',',index=Fa
lse, header=False)
test_data.astype(float).to_csv('kc_house_test_data.csv',sep=',',index=Fals
e, header=False)
```

4. Модели регрессии

4.1. Линейная регрессия

Модель **линейной регрессии** имеет вид:

$price = w_0 + \sum_{j=1}^{m} w_j x^j + \varepsilon,$

где x^j - значение признака j.

Задача: найти коэффициенты w наиболее точно предсказывающие цены на имеющихся данных. Как измерить ошибку?

$$\varepsilon_i = price_i - w_0 - \sum_{j=1}^{m} w_j x_i^j$$

Определение **коэффициентов регрессии**

Введем обозначения:

$$\tilde{X} = (\tilde{x}_{ij}) = \begin{pmatrix} 1 & x_1^1 & \dots & x_1^m \\ \dots & \dots & \dots & \dots \\ 1 & x_n^1 & \dots & x_n^m \end{pmatrix}, w = \begin{pmatrix} w_0 \\ \dots \\ w_m \end{pmatrix}, y = \begin{pmatrix} y_1 \\ \dots \\ y_n \end{pmatrix}$$

В случае, когда матрица $\tilde{X}^T\tilde{X}$ невырождена, система нормальных уравнений имеет единственное решение:

$$\widehat{w} = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T y$$

Каждый коэффициент регрессии отражает влияние конкретного признака на цену.

Если матрица $\tilde{X}^T\tilde{X}$ вырождена, система может не иметь решений или иметь бесконечное множество решений. Нет возможности интерпретировать коэффициенты регрессии.

Как быть, когда матрица вырождена или близка к вырожденной (плохо обусловленные или некорректные задачи)?

Значимые коэффициенты регрессии

Прогнозируемая переменная $Y = w_0 + \sum_{j=1}^{m} w_j x^j + \varepsilon,\ \varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon)$, а коэффициенты регрессии, найденные с помощью МНК, равны $\widehat{w_0}, \dots, \widehat{w_m}$,. Как понять, что $w_j \neq 0$?

Z-score:

$$z_i = \frac{\widehat{w_i}}{\hat{\sigma}\sqrt{v_i}},\ \forall i = \overline{0, m}$$

где $\hat{\sigma} = \sqrt{\frac{1}{n-m-1}\sum_{i=1}^{n}(y_i - \widehat{w_0} - \sum_{j=1}^{m}\widehat{w_j}x^j)^2}$, v_i - диагональный элемент $(\tilde{X}^T\tilde{X})^{-1}$.

Если принять за нулевую гипотезу, что $w_j = 0$, то:

$$z_i \sim t_{n-m-1}$$

Чем больше абсолютное значение z_i, тем больше уверенность, что $w_j \neq 0$.

Модель линейной регрессия в Scikit-learn

```
from sklearn import linear_model

def linear_regression_model(dataX, dataY):
# Создать линейную регрессионную модель normalize=True
regr = linear_model.LinearRegression()
regr.fit(dataX, dataY) # Обучениемодели
return regr
```

Прогнозирование (Scikit-learn)

```
regressionModel.predict(data)
```

Коэффициент детерминации

Коэффициент детерминации. Коэффициент детерминации определяется следующим образом:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

где $\bar{y}$ - среднее значение по наблюдаемым данным, $\widehat{y_i}$ - значения объясняемой переменной, рассчитанные с помощью функции регрессии.

Коэффициент детерминации принимает значение от 0 до 1.

Для нахождения R^2 необходимо вызвать функцию r2_score

```
from sklearn.metrics import r2_score
r2_score (trueY, predictedY)
```

Среднеквадратичная ошибка

Для нахождения среднеквадратичной ошибки определим функцию *RMSE*.

Команда *mean_squared_error*, возвращает средний квадрат ошибки (MSE).

```
from sklearn.metrics import mean_squared_error
def RMSE(trueY, predictedY):
    return np.sqrt(mean_squared_error(trueY, predictedY))
```

Качество регрессии (kc_house_data)

- Как влияет формат данных на качество регрессии?
- Действительно ли стоимость дома линейно зависит от признаков bedrooms, bathrooms, floors, view, condition, grade?

Будем считать эти признаки категориальными и бинаризуем их.

```
categorial_cols = ['floors', 'view', 'condition', 'grade', 'bedrooms',
'bathrooms','zipcode']
for cc in categorial_cols:
    dummies = pd.get_dummies(data[cc], drop_first=False)
    dummies = dummies.add_prefix("{}#".format(cc))
data.drop(cc, axis=1, inplace=True)
    data = data.join(dummies)
print(data.shape)
for column in data:
print(column,end=';')
print()
```

В результате был получен набор с 79 признаками.

4.2. Гребневая регрессия

Гребневая регрессия также является линейной моделью регрессии, поэтому ее формула аналогична той, что используется в обычном методе наименьших квадратов. В гребневой регрессии коэффициенты (w) выбираются не только с точки зрения того, насколько хорошо они позволяют предсказывать на обучающих данных, они еще подгоняются в соответствии с дополнительным ограничением. Нам нужно, чтобы величина коэффициентов была как можно меньше. Другими словами, все элементы w должны быть близки к нулю. Это означает, что каждый признак должен иметь как можно меньшее влияние на результат (то есть каждый признак должен иметь небольшой регрессионный коэффициент) и в то же время он должен по-прежнему обладать хорошей прогнозной силой. Это ограничение является примером регуляризации (regularization). Регуляризация означает явное ограничение модели для предотвращения переобучения. Регуляризация, использующаяся в гребневой регрессии, известна как L2 регуляризация.

Метод гребневой регрессии решает проблему вырожденности с помощью добавления к функционалу Q регуляризатора, штрафующего большие значения квадрата евклидовой нормы вектора w:

$$Q_{\text{гр}}(w_0, \dots, w_m) := \left\|\tilde{X}w - y\right\|_2^2 + \alpha\|w\|_2^2 \to \min_w$$

где $\alpha \geq 0$

Для дифференцируемой функции $Q_{\text{гр}}$ необходимое условие минимума $\frac{\partial Q_{\text{гр}}}{\partial w_j} = 0, j = 0,1,\dots,m$ в матричной форме имеет вид:

$$2\tilde{X}^T\left(\tilde{X}\hat{w} - y\right) + 2\alpha I_n\hat{w} = 0 \Longrightarrow (\tilde{X}^T\tilde{X} + \alpha I_n)\hat{w} = \tilde{X}^T y$$

Откуда получаем, что, в случае невырожденности матрицы $\tilde{X}^T\tilde{X} + \alpha I_n$, решение системы нормальных уравнений выглядит следующим образом:

$$\hat{w} = (\tilde{X}^T\tilde{X} + \alpha I_n)^{-1}\tilde{X}^T y$$

Гребневая регрессии (Scikit-learn)

Гребневая регрессии реализована в классе linear_model.Ridge.

```
def ridge_regression_model(dataX, dataY, alphaParam):
# Инициализациямодели
ridge = linear_model.Ridge(alpha= alphaParam)
ridge.fit(dataX, dataY) # Обучение модели
return ridge
```

Модель Ridge позволяет найти компромисс между простотой модели (получением коэффициентов, близких к нулю) и качеством ее работы на обучающем наборе. Компромисс между простотой модели и качеством работы на обучающем наборе может быть задан пользователем при помощи параметра alpha. Оптимальное значение alpha зависит от конкретного используемого набора данных. Увеличение alpha заставляет коэффициенты сжиматься до близких к нулю значений, что снижает качество работы модели на обучающем наборе, но может улучшить ее обобщающую способность.

4.3. Лассо-регрессия

Альтернативой Ridge как метода регуляризации линейной регрессии является Lasso (англ.Least Absolute Shrinkage and Selection Operator, Оператор наименьшего сжатияи выбора). Как и гребневая регрессия, лассо также сжимает коэффициенты до близких к нулю значений, но несколько иным способом, называемым L1 регуляризацией. Результат L1 регуляризации заключается в том, что при использовании Лассо некоторые коэффициенты становятся равны точно нулю. Получается, что некоторые признаки полностью исключаются из модели. Это можно рассматривать как один из видов автоматического отбора признаков. Получение нулевых значений для некоторых коэффициентов часто упрощает интерпретацию модели и может выявить наиболее важные признаки вашей модели.

Метод Лассо штрафует большие значения L_1- нормы вектора w:

$$Q_{\text{л}}(w_0,\dots,w_m) \coloneqq \left\|\tilde{X}w - y\right\|_2^2 + \beta\|w\|_1 \to \min_w$$

где $\beta \geq 0$

Свойство:

При увеличении β количество коэффициентов регрессии равных нулю увеличивается, то есть происходит отбор значимых признаков (featureselection).

Функция $Q_{\text{л}}$ не является дифференцируемой функцией на всем множестве $\mathbb{R}^{m+1}$, но является субдифференцируемой, поэтому воспользуемся необходимым условием минимума в терминах субдифференциала:

$$0 \in \partial_{w_j} Q_{\text{л}}, j = 0,1,\dots,m$$

Подробно распишем чему равен $\partial_{w_j} Q_{\text{л}}$:

$$\partial_{w_j} Q_л = \frac{\partial Q}{\partial w_j} + \beta \partial_{w_j} |w_j| = -2\rho_j + 2z_j w_j + \begin{cases} -\beta, \text{ при } w_j < 0 \\ [-\beta, \beta], \text{ при } w_j = 0 \\ \beta, \text{ при } w_j < 0 \end{cases}$$

$$= \begin{cases} 2z_j w_j - 2\rho_j - \beta, \text{ при } w_j < 0 \\ [-2\rho_j - \beta, -2\rho_j + \beta], \text{ при } w_j = 0 \\ 2z_j w_j - 2\rho_j + \beta, \text{ при } w_j < 0 \end{cases}$$

где $\rho_j = \sum_{i=1}^{n} \tilde{x}_{ij}\left(y_i - \sum_{k=0.k\neq j}^{m} w_k \tilde{x}_{ik}\right), z_j = \sum_{i=1}^{n} \left(\tilde{x}_{ij}\right)^2$

Получаем, что $\hat{w}_j = \begin{cases} (\rho_j + \beta/2)/z_j, \text{ при } \rho_j < -\beta/2 \\ 0, \text{ при } \rho_j \in [-\beta/2, \beta/2] \\ (\rho_j - \beta/2)/z_j, \text{ при } \rho_j > \beta/2 \end{cases}$

В Scikit-learn для нахождения коэффициентов регрессии методом Лассо используется следующий функционал качества:

$$\frac{1}{2n} \left\| \tilde{X} w - y \right\|_2^2 + \alpha \|w\|_1 \to \min_w$$

где$\alpha \geq 0$

Лассо (Scikit-*learn*)

```
def lasso_model(dataX, dataY, alphaParam):
# Инициализациямодели
lasso = linear_model.Lasso(alpha=alphaParam, max_iter=1000)
lasso.fit(dataX, dataY)  # Обучение модели
return lasso
```

На практике, когда стоит выбор между гребневой регрессией и лассо, предпочтение, как правило, отдается гребневой регрессии. Однако, если у вас есть большое количество признаков и есть основания считать, что лишь некоторые из них важны, Lasso может быть оптимальным выбором. Аналогично, если вам нужна легко интерпретируемая модель, Lasso поможет получить такую модель, так как она выберет лишь подмножество входных признаков.

5. Практическое задание

1. Выполните предобработку данных (preprocessing):
   а) Анализ и удаление выбросов
   б) Анализ и восстановление пропусков
   в) Стандартизация данных
   г) Обсудить возможность выделения характерных признаков (featureextraction).
2. Выполнить вычисления по модели многомерной линейной регрессии и провести анализ полученной модели:
   а) Оценить качество регрессии по коэффициенту детерминации
   б) Оценить ошибку RMSE
   в) Выделить значимые и незначимые коэффициенты регрессии

Качество регрессии

	RMSE	R^2
Линейная регрессия		
Гребневая регрессия($\alpha = 3$)		
Лассо ($\alpha = 120$)		

Коэффициенты регрессии

	Линейная регрессия	Гребневая регрессия ($\alpha = 3$)	Лассо ($\alpha = 120$)
sqft_living			
sqft_lot			
waterfront			
sqft_above			
yr_built			
yr_renovated			
sqft_living15			
sqft_lot15			
floors#1.0			
floors#1.5			
floors#2.0			
floors#2.5			
floors#3.0			
floors#3.5			