Методы машинного обучения.

Лекция №7

Дементьева Кристина Игоревна,
старший преподаватель кафедры Прикладной математики и кибернетики СибГУТИ

kirsten-frost@mail.ru

Линейная регрессия

Регрессионный анализ — набор статистических методов исследования влияния одной или нескольких независимых переменных X_1, X_2, X_3… на зависимую переменную Y.

Линейная регрессия — это алгоритм, который пытается применить отношения, которые будут предсказывать исход события на основе точек данных независимой переменной. Отношение обычно представляет собой прямую линию, которая наилучшим образом соответствует различным точкам данных как можно ближе. Выходные данные имеют непрерывную форму, т. е. числовое значение. Например, результатом может быть доход или продажи в валюте, количество проданных продуктов и т. д.

Линейная регрессия

Целевая функция линейной регрессионной модели:

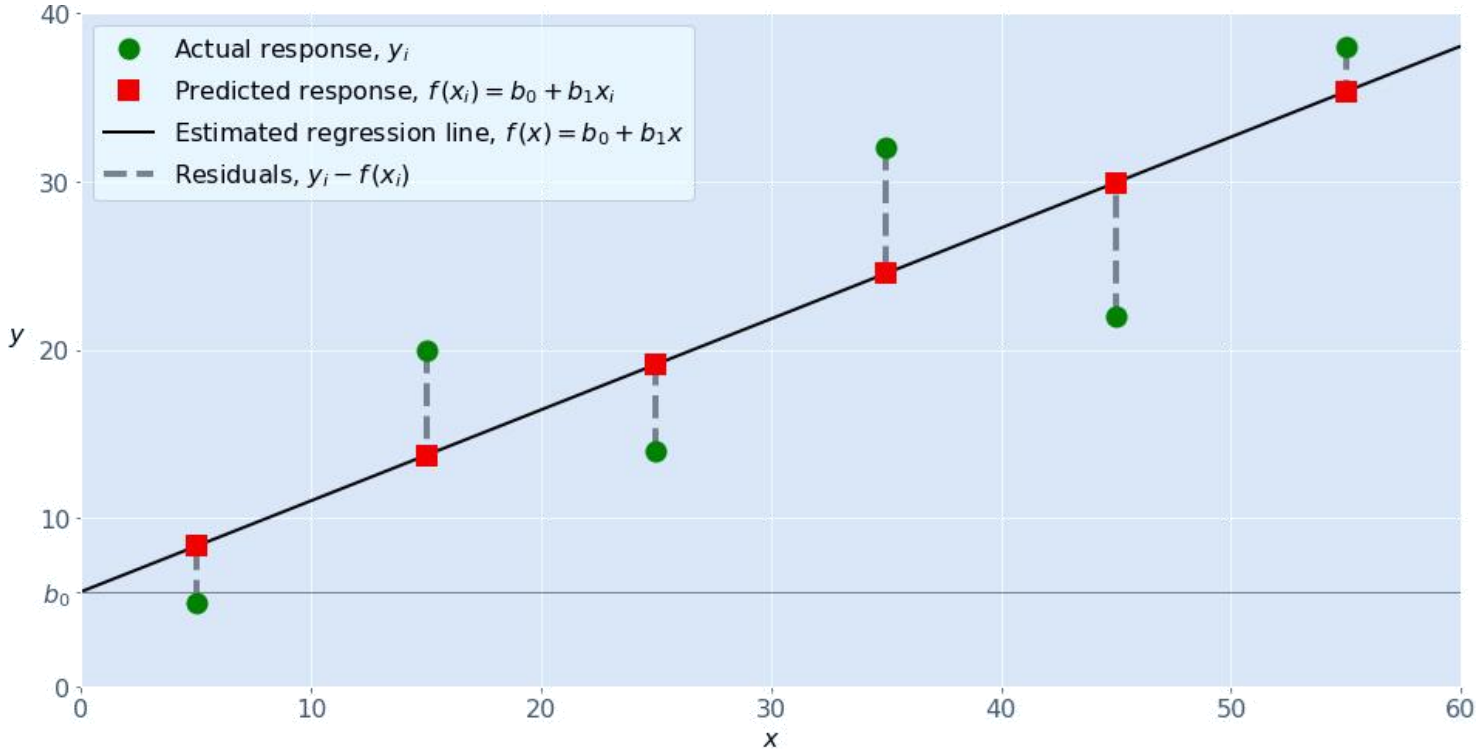
$$y=f(x,b)+\varepsilon$$

где b — параметры модели,
ε — случайная ошибка модели

$f(x,b)$ имеет вид $f(x,b)= b_0 + b_1x_1 + b_2x_2 + \cdots + b_kx_k$,

где — b_j параметры (коэффициенты) регрессии,
x_j — регрессоры (факторы модели),
k — количество факторов модели.

Линейная регрессия

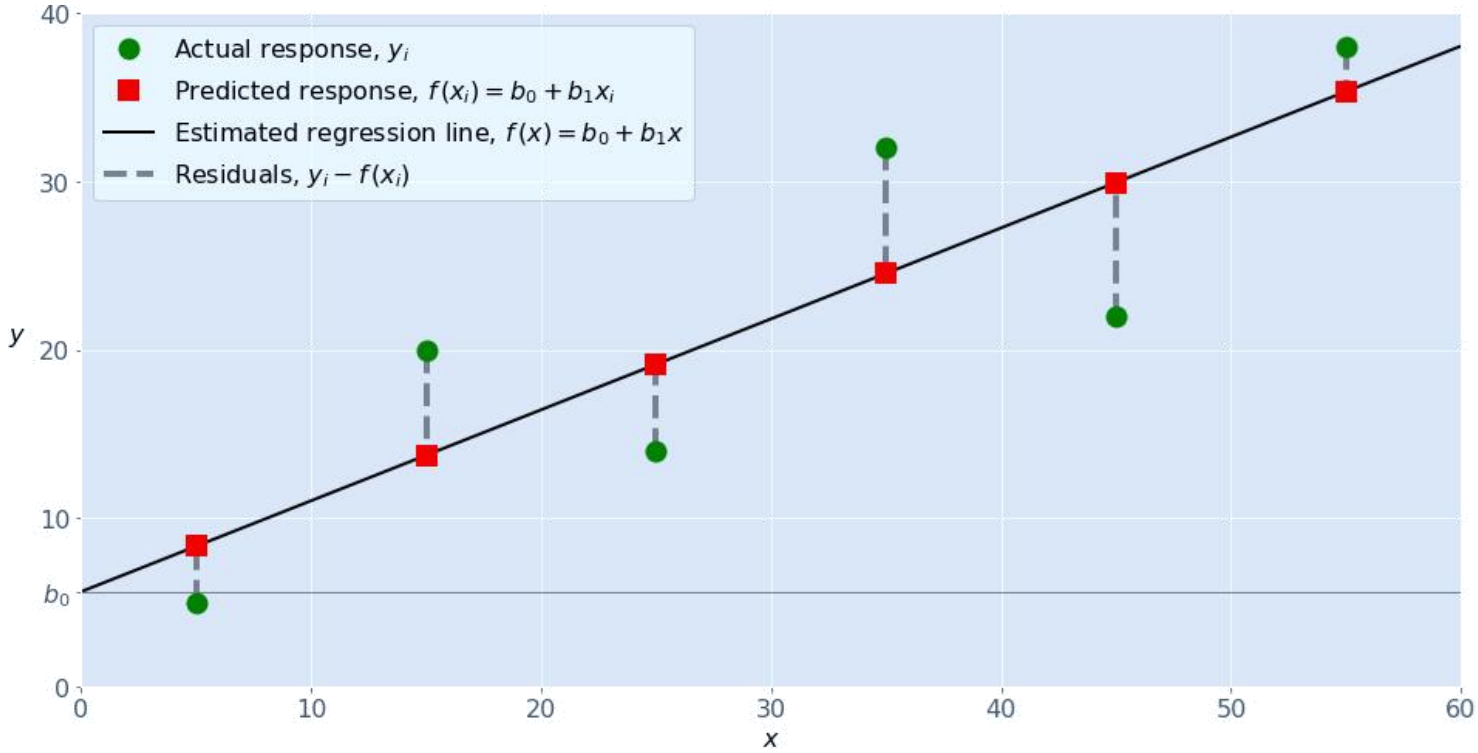


Зелёные круги - заданный набором пар входов-выходов (x-y).

Эти пары – результаты наблюдений.
Например, наблюдение, крайнее слева (зелёный круг) имеет на входе x = 5 и соответствующий выход (ответ) y = 5.

Оценочная функция регрессии (чёрная линия) выражается уравнением $f(x) = b_0 + b_1 x$.
Нужно рассчитать оптимальные значения спрогнозированных весов b_0 и b_1 для минимизации SSR и определить оценочную функцию регрессии.
b_0 показывает точку, где расчётная линия регрессии пересекает ось y.
Величина b_1 определяет наклон расчетной линии регрессии.

Линейная регрессия



Предсказанные ответы (красные квадраты) – точки линии регрессии, соответствующие входным значениям.

Для входа x = 5 предсказанный ответ равен f(5) = 8.33 (представленный крайним левыми квадратом).

Остатки (вертикальные пунктирные серые линии) вычисляются как $y_i - f(x_i) = y_i - b_0 - b_1 x_i$ для i = 1, …, n. Они представляют собой расстояния между зелёными и красными пунктами. При реализации линейной регрессии нужно минимизировать эти расстояния и сделать красные квадраты как можно ближе к предопределённым зелёным кругам.

Линейная регрессия

```
from sklearn.linear_model import LinearRegression
```

Несколько опциональных параметров класса LinearRegression:

fit_intercept – логический (True по умолчанию) параметр, который решает, вычислять отрезок b_0 (True) или рассматривать его как равный нулю (False).
normalize – логический (False по умолчанию) параметр, который решает, нормализовать входные переменные (True) или нет (False).
copy_X – логический (True по умолчанию) параметр, который решает, копировать (True) или перезаписывать входные переменные (False).
n_jobs – целое или None (по умолчанию), представляющее количество процессов, задействованных в параллельных вычислениях. None означает отсутствие процессов, при -1 используются все доступные процессоры.

Линейная регрессия

```
import numpy as np
from sklearn.linear_model import LinearRegression

x = np.array([5, 15, 25, 35, 45, 55]).reshape((-1, 1))
y = np.array([5, 20, 14, 32, 22, 38])
model = LinearRegression()
model = LinearRegression().fit(x, y)
```

Линейная регрессия

model содержит атрибуты ***.intercept_***, который представляет собой коэффициент b_0, и ***.coef_***, который представляет b_1:

```
print('intercept:', model.intercept_)
print('slope:', model.coef_)
```

```
intercept: 5.633333333333329
slope: [0.54]
```

Примерное значение b_0 = 5.63 показывает, что модель предсказывает ответ 5.63 при x, равном нулю.
Равенство b_1 = 0.54 означает, что предсказанный ответ возрастает до 0.54 при x, увеличенным на единицу.

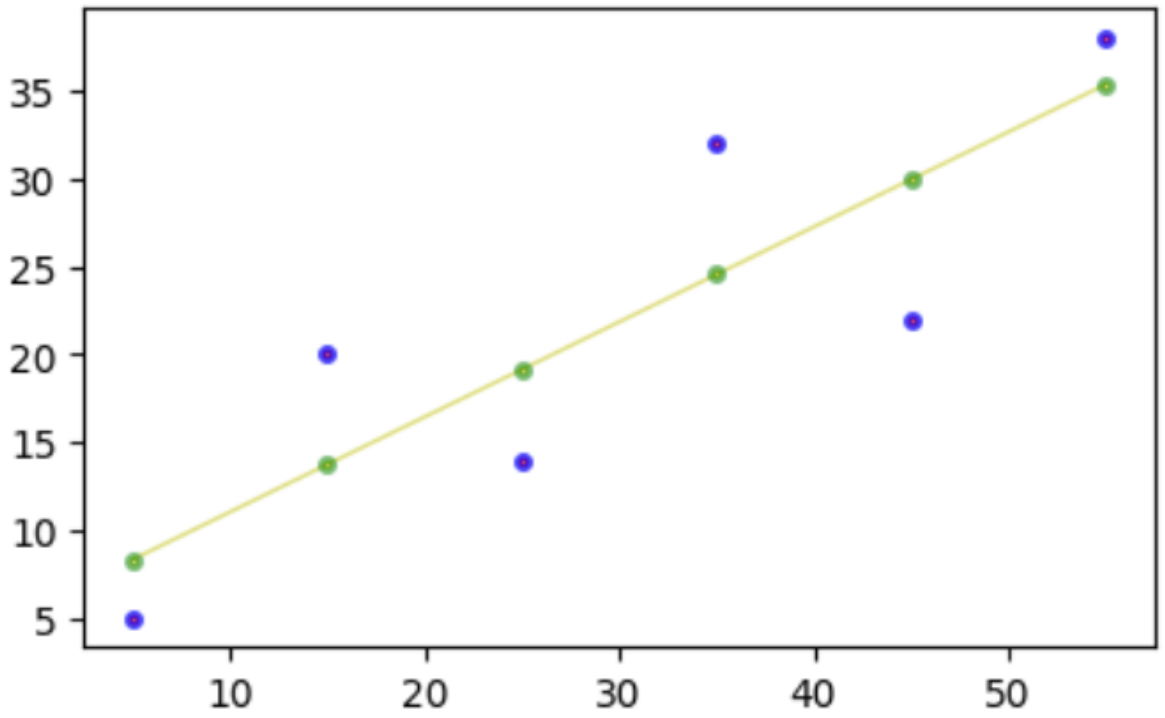
Линейная регрессия

```
y_pred = model.predict(x)
print('predicted:', y_pred, sep='\n')
```

```
predicted:
[ 8.33333333 13.73333333 19.13333333 24.53333333 29.93333333 35.33333333]
```

Линейная регрессия

```
import matplotlib.pyplot as plt
plt.figure(figsize=(5, 3))
plt.plot(x, y, 'o-r', alpha=0.7, lw=0, mec='b', mew=2, ms=3)
plt.plot(x, y_pred, 'o-y', alpha=0.5, lw=1, mec='g', mew=2, ms=3)
```



Оценка результатов

```python
import numpy as np
from sklearn.metrics import mean_squared_error, mean_absolute_error

y_true = [3, -0.5, 2, 7]
y_pred = [2.5, 0.0, 2, 8]

print(f"MSE: {mean_squared_error(y_true, y_pred):.2f}")
print(f"MAE: {mean_absolute_error(y_true, y_pred):.2f}")
```

```
MSE: 0.38
MAE: 0.50
```

Оценка результатов

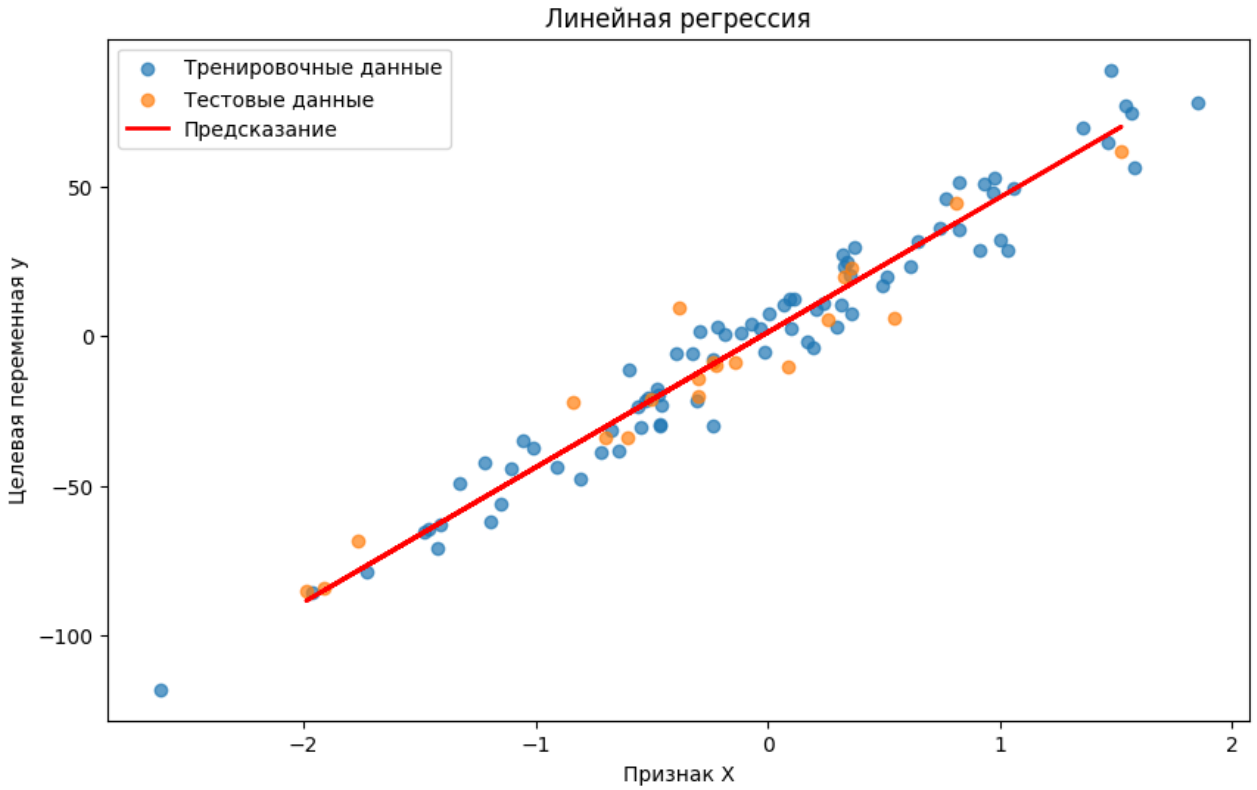
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_regression

X, y = make_regression(n_samples=100, n_features=1, noise=10)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

Генерация данных для регрессии

Создание и обучение модели

Визуализация результатов

```
plt.figure(figsize=(10, 6))
plt.scatter(X_train, y_train, alpha=0.7, label='Тренировочные данные')
plt.scatter(X_test, y_test, alpha=0.7, label='Тестовые данные')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Предсказание')
plt.xlabel('Признак X')
plt.ylabel('Целевая переменная y')
plt.legend()
plt.title('Линейная регрессия')
plt.show()
```



Множественная регрессия

На данных о ценах на жилье в Калифорнии

```
from sklearn.datasets import fetch_california_housing
import pandas as pd

housing = fetch_california_housing()
X, y = housing.data, housing.target
feature_names = housing.feature_names

df = pd.DataFrame(X, columns=feature_names)
df['Price'] = y

print("Первые 5 строк данных:")
print(df.head())

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = LinearRegression()
model.fit(X_train, y_train)

coefficients = pd.DataFrame({
    'Признак': feature_names,
    'Коэффициент': model.coef_
})
print("\nКоэффициенты модели:")
print(coefficients)
```

```
Первые 5 строк данных:
   MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  Longitude  Price
0  8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88    -122.23  4.526
1  8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86    -122.22  3.585
2  7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85    -122.24  3.521
3  5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85    -122.25  3.413
4  3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85    -122.25  3.422
```

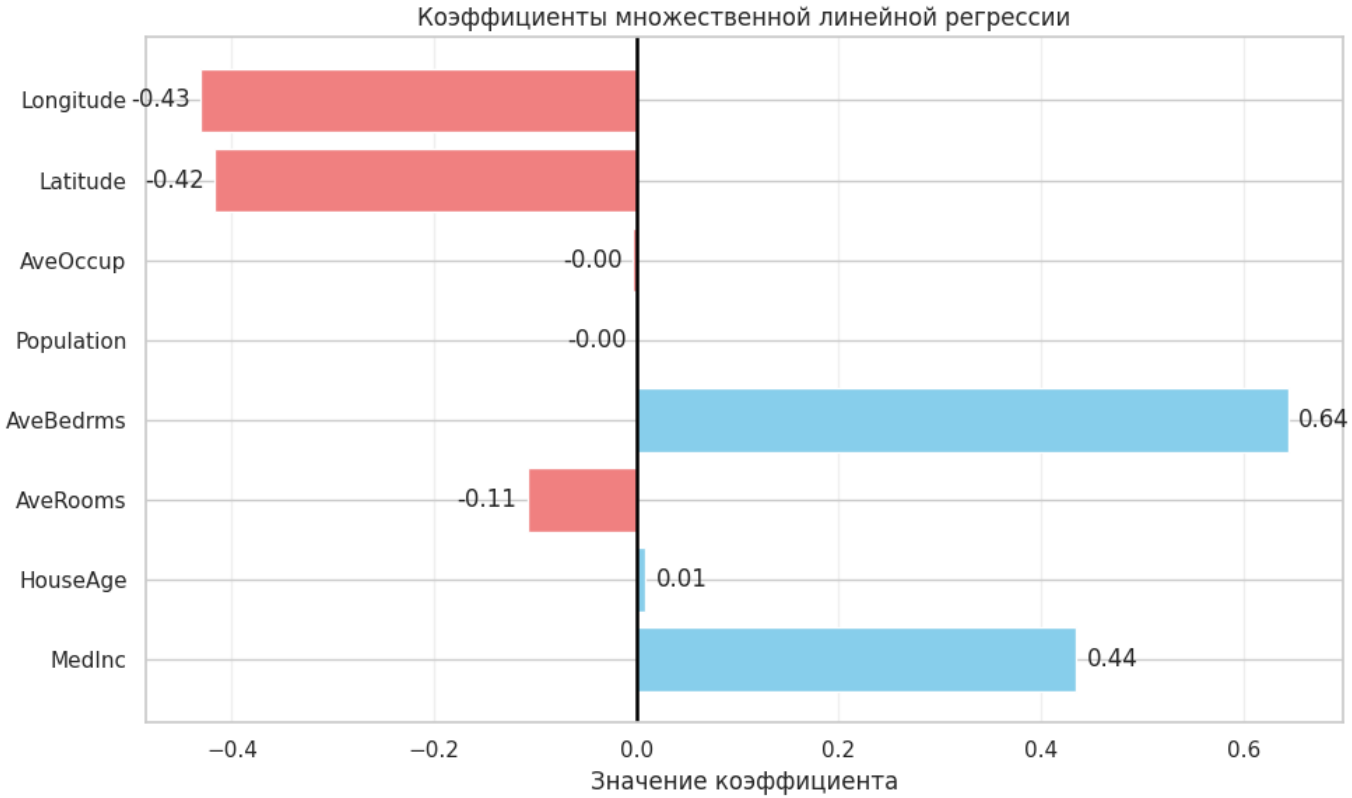
```
Коэффициенты модели:
      Признак  Коэффициент
0      MedInc     0.435001
1    HouseAge     0.009250
2    AveRooms    -0.108041
3   AveBedrms     0.644489
4  Population    -0.000006
5    AveOccup    -0.003424
6    Latitude    -0.416792
7   Longitude    -0.431204
```

Множественная регрессия

На данных о ценах на жилье в Калифорнии

Коэффициенты модели:

	Признак	Коэффициент
0	MedInc	0.435001
1	HouseAge	0.009250
2	AveRooms	-0.108041
3	AveBedrms	0.644489
4	Population	-0.000006
5	AveOccup	-0.003424
6	Latitude	-0.416792
7	Longitude	-0.431204



$$f(x,b) = b_0 + b_1x_1 + b_2x_2 + \cdots + b_kx_k$$

Регрессия LASSO

```
from sklearn.linear_model import Lasso
```

Лассо — это линейная модель, которая оценивает разреженные коэффициенты. Это полезно в некоторых контекстах из-за своей тенденции отдавать предпочтение решениям с меньшим количеством ненулевых коэффициентов, эффективно уменьшая количество функций, от которых зависит данное решение.

```
y_pred = model.predict(x)
print('predicted:', y_pred, sep='\n')
```

```
predicted:
[ 8.41904762 13.7847619  19.15047619 24.51619048 29.88190476 35.24761905]
```

Регрессия LASSO

Параметр *alpha* контролирует степень потенциальности предполагаемых коэффициентов.

```
import numpy as np
from sklearn.linear_model import Lasso

x = np.array([5, 15, 25, 35, 45, 55]).reshape((-1, 1))
y = np.array([5, 20, 14, 32, 22, 38])

for i in range(1, 11):
  model = Lasso(alpha = i *0.1).fit(x, y)
  y_pred = model.predict(x)
  print('alpha=', round(i*0.1, 1), ': ', y_pred)
```

```
alpha= 0.1 :  [ 8.34190476 13.73847619 19.13504762 24.53161905 29.92819048 35.3247619 ]
alpha= 0.2 :  [ 8.35047619 13.74361905 19.1367619  24.52990476 29.92304762 35.31619048]
alpha= 0.3 :  [ 8.35904762 13.7487619  19.13847619 24.52819048 29.91790476 35.30761905]
alpha= 0.4 :  [ 8.36761905 13.75390476 19.14019048 24.52647619 29.9127619  35.29904762]
alpha= 0.5 :  [ 8.37619048 13.75904762 19.14190476 24.5247619  29.90761905 35.29047619]
alpha= 0.6 :  [ 8.3847619  13.76419048 19.14361905 24.52304762 29.90247619 35.28190476]
alpha= 0.7 :  [ 8.39333333 13.76933333 19.14533333 24.52133333 29.89733333 35.27333333]
alpha= 0.8 :  [ 8.40190476 13.77447619 19.14704762 24.51961905 29.89219048 35.2647619 ]
alpha= 0.9 :  [ 8.41047619 13.77961905 19.1487619  24.51790476 29.88704762 35.25619048]
alpha= 1.0 :  [ 8.41904762 13.7847619  19.15047619 24.51619048 29.88190476 35.24761905]
```

Регрессия LARS

```
from sklearn.linear_model import Lars
```

Регрессия наименьшего угла (**LARS**) — это алгоритм регрессии для многомерных данных. LARS похож на пошаговую регрессию вперед. На каждом этапе он находит функцию, наиболее коррелирующую с целью. Когда есть несколько объектов, имеющих одинаковую корреляцию, вместо того, чтобы продолжать движение по одному и тому же объекту, он движется в одинаковом направлении между объектами.

Регрессия ElasticNet

```python
import numpy as np
from sklearn.linear_model import ElasticNet

x = np.array([5, 15, 25, 35, 45, 55]).reshape((-1, 1))
y = np.array([5, 20, 14, 32, 22, 38])

for i in range(1, 11):
  model = ElasticNet(alpha=i*0.1).fit(x, y)
  y_pred = model.predict(x)
  print('alpha=', round(i*0.1, 1), ': ', y_pred)
```

```
alpha= 0.1 :  [ 8.3399322  13.73729265 19.13465311 24.53201356 29.92937401 35.32673446]
alpha= 0.2 :  [ 8.34652881 13.74125062 19.13597243 24.53069424 29.92541605 35.32013786]
alpha= 0.3 :  [ 8.35312316 13.74520723 19.1372913  24.52937537 29.92145944 35.31354351]
alpha= 0.4 :  [ 8.35971524 13.74916248 19.13860972 24.52805695 29.91750419 35.30695142]
alpha= 0.5 :  [ 8.36630507 13.75311638 19.13992768 24.52673899 29.91355029 35.30036159]
alpha= 0.6 :  [ 8.37289264 13.75706892 19.1412452  24.52542147 29.90959775 35.29377402]
alpha= 0.7 :  [ 8.37947796 13.76102011 19.14256226 24.52410441 29.90564656 35.28718871]
alpha= 0.8 :  [ 8.38606102 13.76496995 19.14387887 24.5227878  29.90169672 35.28060565]
alpha= 0.9 :  [ 8.39264183 13.76891843 19.14519503 24.52147163 29.89774824 35.27402484]
alpha= 1.0 :  [ 8.39922038 13.77286556 19.14651074 24.52015592 29.8938011  35.26744628]
```

Более устойчива к переобучению, чем обычная линейная регрессия.

Может автоматически отбирать важные предикторы.

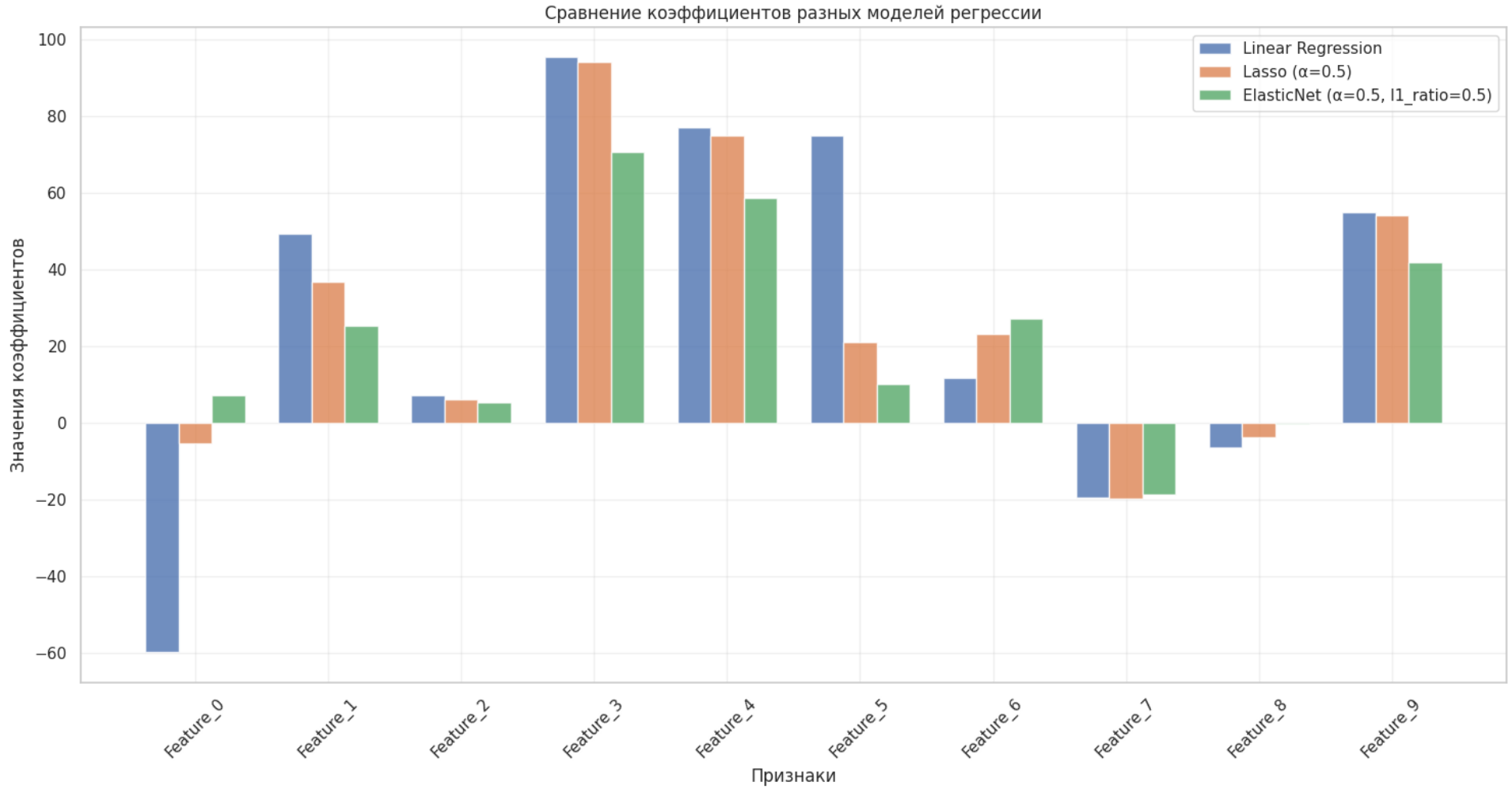
Регрессия ElasticNet

Метод **ElasticNetCV** можно использовать для установки параметров *alpha* и *l1_ratio* путем перекрестной проверки.

```
model = ElasticNetCV().fit(x, y)
y_pred = model.predict(x)
print( y_pred)
```

```
[ 8.95792979 14.1080912  19.25825262 24.40841404 29.55857546 34.70873688]
```

Сравнение коэффициентов разных моделей регрессии



X, y = make_regression(n_samples=100, n_features=10, noise=20)

Сравнение коэффициентов разных моделей регрессии