МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА СИСТЕМ СБОРА И ОБРАБОТКИ ДАННЫХ

НГТУ
НЭТИ

Лабораторная работа №1

по дисциплине «Машинное обучение»

на тему

«Регрессионный анализ данных методами машинного обучения»

Группа: АТМ-25

Факультет: АВТФ

Студент: Секачёв Г. М.

Преподаватель: Павлова А. И.

Новосибирск, 2025

СОДЕРЖАНИЕ

1. Цель работы

Построить модели линейной, полиномиальной (полином 2 или 3 степени), логистической и ридж-регрессии для прогнозирования средней стоимости домов в пригородах Бостона.

Целевая переменная: medv – средняя стоимость домов (в тысячах долларов).

Признаки:

- crim: уровень преступности на душу населения,
- zn: доля земли, предназначенной под жилые постройки,
- indus: доля нефункционирующих промышленных предприятий на городскую территорию,
- chas: бинарный признак близости реки Чарльз (1 – река рядом, 0 – нет),
- nox: концентрация оксида азота,
- rm: среднее количество жилых комнат в доме,
- age: средний возраст зданий,
- dis: расстояние до пяти крупных центров занятости,
- rad: доступность автомобильных дорог,
- tax: ставка налога на недвижимость,
- ptratio: соотношение учеников к учителям в школах региона,
- b: отношение числа афроамериканцев к общей численности населения,
- lstat: процент бедного населения в регионе.

2. Обработка данных

Датасет не содержит NA или нечисловых элементов.

```
df = pd.read_csv('BostonHousing_raw.csv', # путь к файлу, (используй автодотолнение)
                 sep=',', # разделитель данных в файле
                 header=0, # номер строки с заголовками, нумерация с нуля
                 # header='None', # если заголовки отсутствуют
                )
print("Размер таблицы", df.shape)
df[:2]
```

```
Размер таблицы (506, 14)
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.9	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6

```
df = df.dropna()
print("Размер таблицы:", df.shape)
```

```
Размер таблицы: (506, 14)
```

3. Обучающая и тестовая выборки

Для обучения модели выберем признаки, которые коррелируют с таргетом хотя бы на 0.4 так как:

1. Некоторые признаки могут быть шумовыми или слабо связанными с целевой переменной. Их удаление улучшает интерпретируемость модели и сокращает вычислительные затраты.
2. Чем меньше признаков, тем быстрее работает алгоритм.
3. Большое количество слабокоррелированных признаков увеличивает риск переобучения, особенно в случаях, когда объем данных невелик.

```
corrmatrix = df.corr()["medv"].abs().sort_values(ascending=False)
print(corrmatrix)
```

```
medv       1.000000
lstat      0.737663
rm         0.695360
ptratio    0.507787
indus      0.483725
tax        0.468536
nox        0.427321
crim       0.388305
rad        0.381626
age        0.376955
zn         0.360445
b          0.333461
dis        0.249929
chas       0.175260
Name: medv, dtype: float64
```

```
t = 0.4
selected_features = corrmatrix[corrmatrix >= t].index.tolist()[1:]
print(selected_features)
```

```
['lstat', 'rm', 'ptratio', 'indus', 'tax', 'nox']
```

Разделим набор данных на обучающую и тестовую выборки. Для обучения используем 80% примеров.

```
x = df[selected_features]
y = df['medv'] # Целевое значение (цена дома)
x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.2, random_state=42
)
```

4. Модели машинного обучения

Создадим 4 модели машинного обучения для регрессионного анализа:

- линейная регрессия,
- регрессия полинома 2 степени,

- регрессия полинома 3 степени,
- ридж-регрессия.

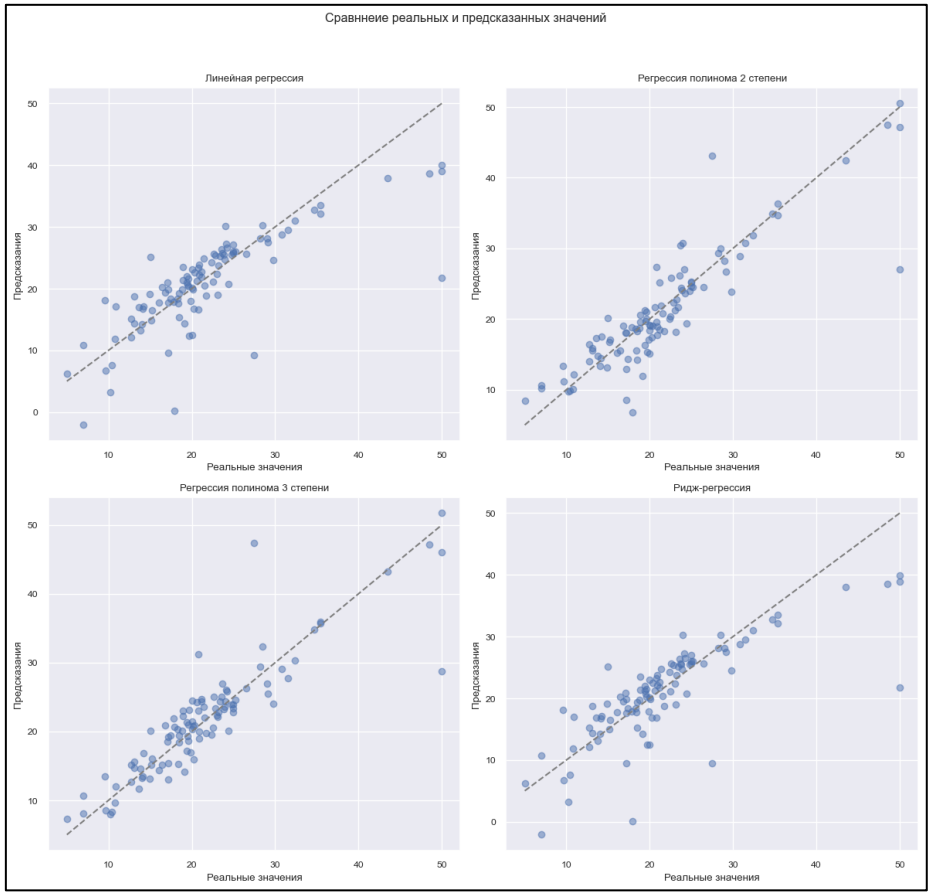
```
# Модель линейной регрессии
lin_reg = LinearRegression()
lin_reg.fit(x_train, y_train)
y_pred_lin = lin_reg.predict(x_test)

# Модель полиномиальной регрессии (степень 2)
poly_reg_2 = make_pipeline(PolynomialFeatures(degree=2), LinearRegression())
poly_reg_2.fit(x_train, y_train)
y_pred_poly_2 = poly_reg_2.predict(x_test)

# Модель полиномиальной регрессии (степень 3)
poly_reg_3 = make_pipeline(PolynomialFeatures(degree=3), LinearRegression())
poly_reg_3.fit(x_train, y_train)
y_pred_poly_3 = poly_reg_3.predict(x_test)

# Ридж-регрессия
ridge_reg = Ridge(alpha=1.0)
ridge_reg.fit(x_train, y_train)
y_pred_ridge = ridge_reg.predict(x_test)
```

5. Прогнозные значения



Из графиков видно, что предсказания довольно близки к реальным значениям у всех моделей. Чуть поменьше шума наблюдается у полиномиальных регрессий, чем у линейной и ридж-регрессии.

6. Оценка точности

```python
# Оценка точности каждой модели
for name, predictions in models:
    r2 = r2_score(y_test, predictions)
    mse = mean_squared_error(y_test, predictions)
    mae = mean_absolute_error(y_test, predictions)
    rmse = np.sqrt(mse)

    print(f'{name}:')
    print(f'  R²: {r2:.4f}')
    print(f'  MSE: {mse:.4f}')
    print(f'  RMSE: {rmse:.4f}')
    print(f'  MAE: {mae:.4f}')
```

```
Линейная регрессия:
  R²: 0.6210
  MSE: 27.7968
  RMSE: 5.2723
  MAE: 3.3543
Регрессия полинома 2 степени:
  R²: 0.7782
  MSE: 16.2667
  RMSE: 4.0332
  MAE: 2.5363
Регрессия полинома 3 степени:
  R²: 0.7975
  MSE: 14.8522
  RMSE: 3.8539
  MAE: 2.4048
Ридж-регрессия:
  R²: 0.6214
  MSE: 27.7641
  RMSE: 5.2692
  MAE: 3.3460
```

- **Полиномиальная регрессия третьей степени** показывает наилучшие результаты среди всех моделей. Она имеет самый высокий R^2 (0.7975), наименьшие MSE (14.8522), RMSE (3.8539) и MAE (2.4048).
- **Полиномиальная регрессия второй степени** также показывает хорошие результаты, но немного уступает третьей степени.
- **Линейная регрессия** и **ридж-регрессия** показывают схожие результаты, которые уступают полиномиальным моделям.

Таким образом, для данной задачи лучше всего подходит полиномиальная регрессия третьей степени.