Министерство цифрового развития

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра прикладной математики и кибернетики

Отчёт

по лабораторной работе № 4 «Сравнительный анализ методов регрессии»

Выполнил:

студент группы ИП-213
Дмитриев Антон Александрович

Работу проверил: Преподаватель
Сороковых Дарья Анатольевна

Новосибирск 2025 г.

Введение (задание)

Задание

Тема: Сравнительный анализ методов регрессии.

Цель: сформировать комплексное понимание различных методов регрессионного анализа и выработать навыки осознанного выбора моделей в зависимости от характеристик данных и решаемой задачи.

Задание

1. Подготовка данных

1.1. Выберите датасет подходящий для решения задачи регрессии с сайта

Kaggle.com.

Датасаты для примера: Egyptian Real Estate Listings, Walmart Sales,

Student Performance , Car Prices Dataset и др.

1.2. Загрузите данные и выполните предобработку:

Обработайте пропущенные значения.

Закодируйте категориальные признаки (Label Encoding).

Разделите данные на обучающую и тестовую выборки.

2. Базовые модели

2.1. Обучите три базовые модели LinearRegression, Lasso и ElasticNet.

3. Оценка качества моделей на тестовой выборке

3.1. Рассчитайте MSE, RMSE, MAE и R^2 для каждой модели

3.2. Постройте графики "предсказанные vs фактические значения"

3.3. Сравните коэффициенты моделей.

4. Подбор гиперпараметров и кросс-валидация

4.1. Реализуйте подбор гиперпараметров для Lasso и ElasticNet.

4.2. Проведите кросс-валидацию для всех моделей.

4.3. Ответьте на вопросы:

- Какие оптимальные гиперпараметры для Lasso и ElasticNet?
- Улучшило ли качество подбор гиперпараметров?

Основная часть

1. Подготовка данных

```
Размер датасета: (19237, 18)

Первые 5 строк:
         ID  Price  Levy Manufacturer    Model  Prod. year   Category  \
0  45654403  13328  1399        LEXUS   RX 450        2010       Jeep
1  44731507  16621  1018    CHEVROLET  Equinox        2011       Jeep
2  45774419   8467     -        HONDA      FIT        2006  Hatchback
3  45769185   3607   862         FORD   Escape        2011       Jeep
4  45809263  11726   446        HONDA      FIT        2014  Hatchback

  Leather interior Fuel type Engine volume    Mileage  Cylinders  \
0              Yes    Hybrid           3.5  186005 km        6.0
1               No    Petrol             3  192000 km        6.0
2               No    Petrol           1.3  200000 km        4.0
3              Yes    Hybrid           2.5  168966 km        4.0
4              Yes    Petrol           1.3   91901 km        4.0

  Gear box type Drive wheels   Doors             Wheel   Color  Airbags
0     Automatic          4x4  04-May        Left wheel  Silver       12
1     Tiptronic          4x4  04-May        Left wheel   Black        8
2      Variator        Front  04-May  Right-hand drive   Black        2
3     Automatic          4x4  04-May        Left wheel   White        0
4     Automatic        Front  04-May        Left wheel  Silver        4

Информация о данных:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19237 entries, 0 to 19236
Data columns (total 18 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ID                19237 non-null  int64
 1   Price             19237 non-null  int64
 2   Levy              19237 non-null  object
 3   Manufacturer      19237 non-null  object
 4   Model             19237 non-null  object
 5   Prod. year        19237 non-null  int64
 6   Category          19237 non-null  object
 7   Leather interior  19237 non-null  object
 8   Fuel type         19237 non-null  object
 9   Engine volume     19237 non-null  object
 10  Mileage           19237 non-null  object
 11  Cylinders         19237 non-null  float64
 12  Gear box type     19237 non-null  object
 13  Drive wheels      19237 non-null  object
 14  Doors             19237 non-null  object
 15  Wheel             19237 non-null  object
 16  Color             19237 non-null  object
 17  Airbags           19237 non-null  int64
dtypes: float64(1), int64(4), object(13)
memory usage: 2.6+ MB
None
```

```
Размеры выборок:
Обучающая: X_train(15389, 17), y_train(15389,)
Тестовая: X_test(3848, 17), y_test(3848,)
```

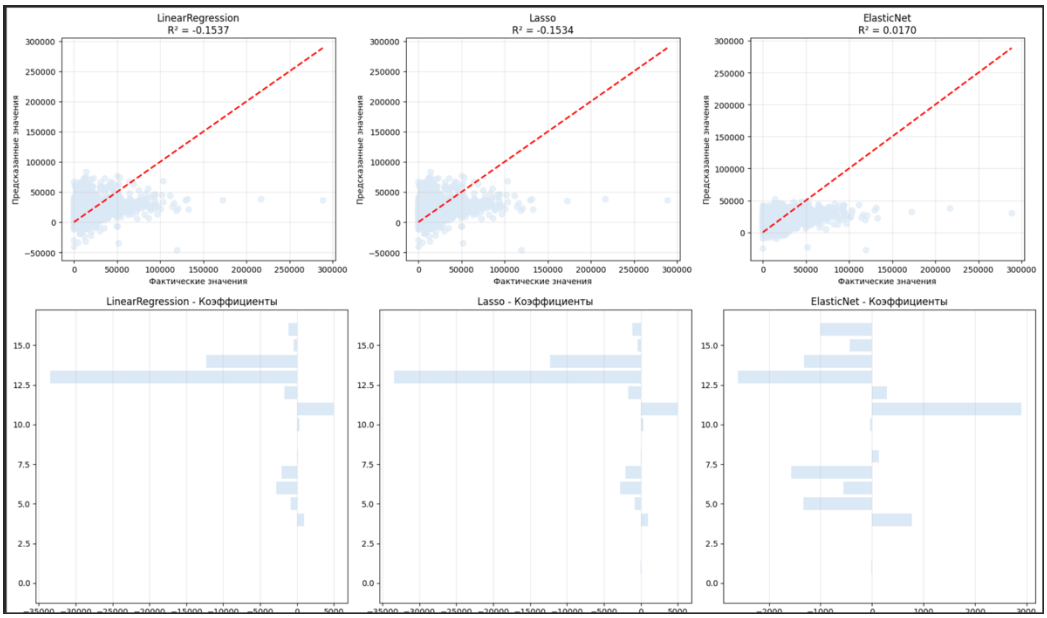
2. Базовые модени

```python
print("\nОБУЧЕНИЕ БАЗОВЫХ МОДЕЛЕЙ")
print("=" * 50)

models = {
    'LinearRegression': LinearRegression(),
    'Lasso': Lasso(random_state=42),
    'ElasticNet': ElasticNet(random_state=42)
}

# Обучение моделей
trained_models = {}
for name, model in models.items():
    print(f"Обучение {name}...")
    model.fit(X_train, y_train)
    trained_models[name] = model
```

```
ОБУЧЕНИЕ БАЗОВЫХ МОДЕЛЕЙ
==================================================
Обучение LinearRegression...
Обучение Lasso...
Обучение ElasticNet...
```

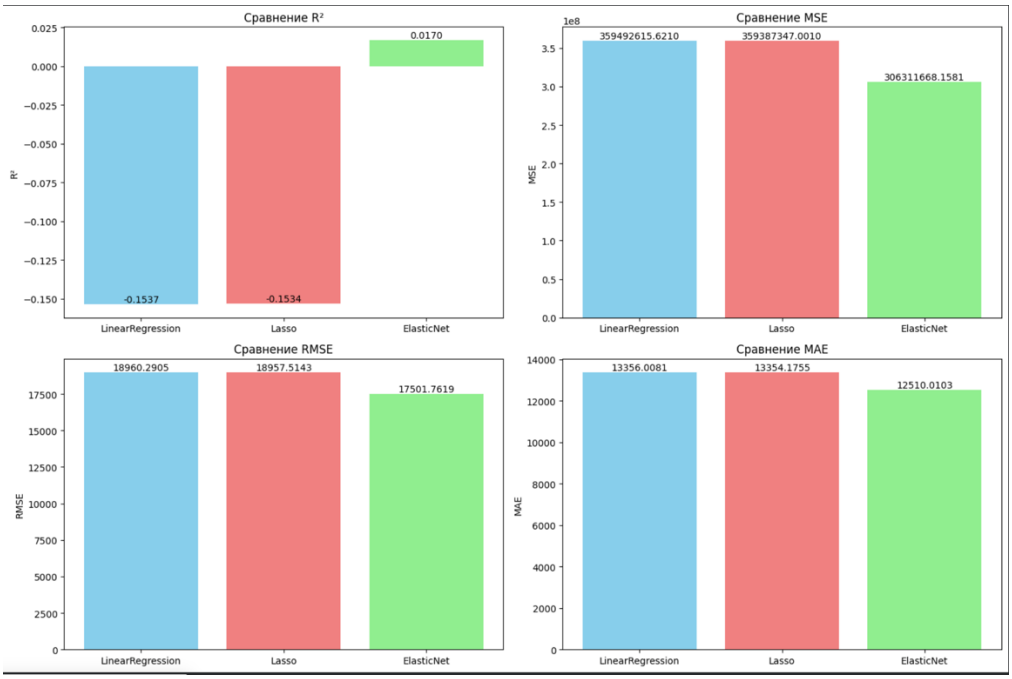
3. Оценка качества модели

```
ОЦЕНКА КАЧЕСТВА МОДЕЛЕЙ НА ТЕСТОВОЙ ВЫБОРКЕ
==================================================

LinearRegression:
MSE: 359492615.62
RMSE: 18960.29
MAE: 13356.01
R²: -0.1537

Lasso:
MSE: 359387347.00
RMSE: 18957.51
MAE: 13354.18
R²: -0.1534

ElasticNet:
MSE: 306311668.16
RMSE: 17501.76
MAE: 12510.01
R²: 0.0170
```



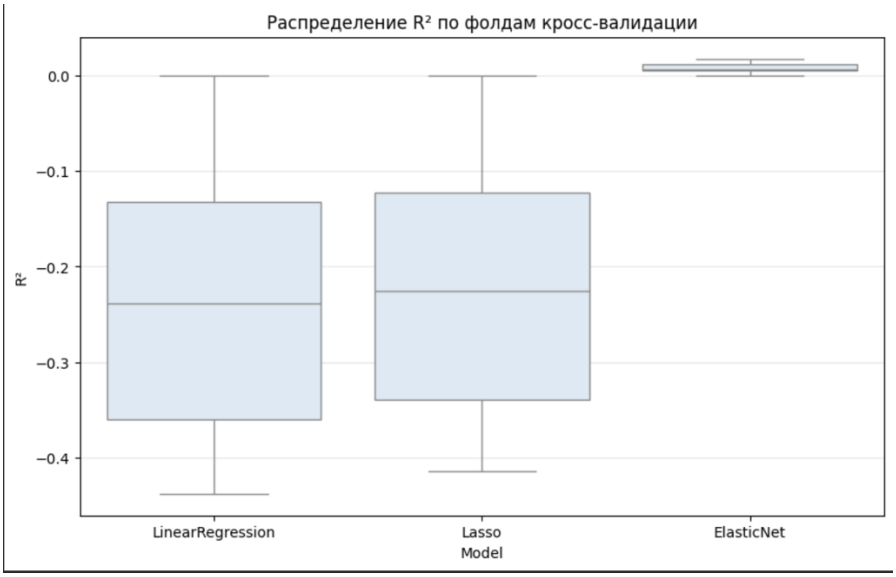
СРАВНЕНИЕ МЕТРИК ВСЕХ МОДЕЛЕЙ:

	MSE	RMSE	MAE	R²
LinearRegression	3.594926e+08	18960.2905	13356.0081	-0.1537
Lasso	3.593873e+08	18957.5143	13354.1755	-0.1534
ElasticNet	3.063117e+08	17501.7619	12510.0103	0.0170

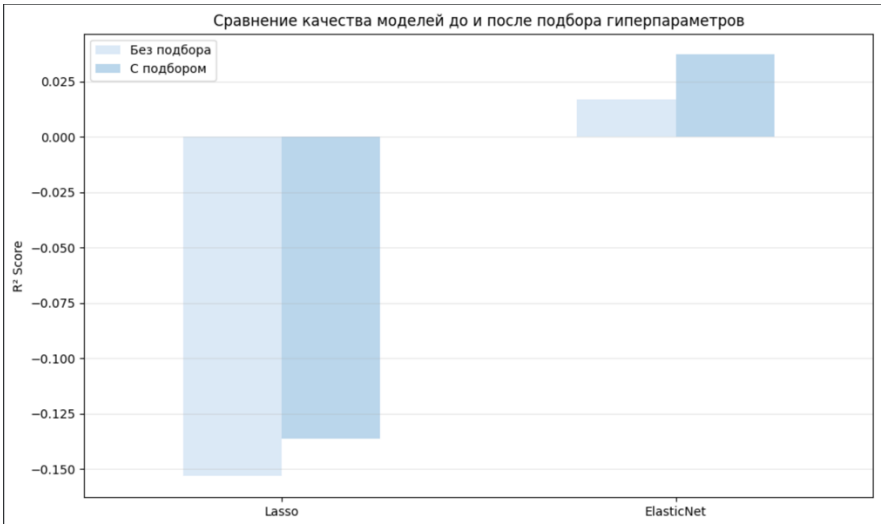


4. Подбор гиперпараметров и кросс-валидация

```
ПОДБОР ГИПЕРПАРАМЕТРОВ И КРОСС-ВАЛИДАЦИЯ
==================================================

Подбор гиперпараметров для Lasso...
Fitting 5 folds for each of 12 candidates, totalling 60 fits
Лучшие параметры для Lasso: {'model__alpha': 100, 'model__max_iter': 1000}
Лучший R² на кросс-валидации: -0.2204

Подбор гиперпараметров для ElasticNet...
Fitting 5 folds for each of 50 candidates, totalling 250 fits
Лучшие параметры для ElasticNet: {'model__alpha': 10, 'model__l1_ratio': 0.7, 'model__max_iter': 1000}
Лучший R² на кросс-валидации: 0.0083
```




```
СРАВНЕНИЕ МОДЕЛЕЙ ПОСЛЕ ПОДБОРА ГИПЕРПАРАМЕТРОВ:
==================================================

Lasso (с подбором гиперпараметров):
R²: -0.1363
RMSE: 18816.49

ElasticNet (с подбором гиперпараметров):
R²: 0.0372
RMSE: 17320.72
```




```
1. Оптимальные гиперпараметры для Lasso и ElasticNet?
   Lasso: {'model__alpha': 100, 'model__max_iter': 1000}
   ElasticNet: {'model__alpha': 10, 'model__l1_ratio': 0.7, 'model__max_iter': 1000}

2. Улучшение качества после подбора гиперпараметров
   Lasso: R² улучшился на 0.0171 (-0.1534 -> -0.1363)
   Относительное улучшение: -11.15%
   ElasticNet: R² улучшился на 0.0202 (0.0170 -> 0.0372)
   Относительное улучшение: 119.29%
```

Заключение

Ссылка на colab:

https://colab.research.google.com/drive/1LQ_mS6vHdnrkZViZzUn9rR37sZi3fBcV?usp=sharing

Код программы

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV,
cross_val_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LinearRegression, Lasso, ElasticNet
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score
from sklearn.pipeline import Pipeline
import warnings
warnings.filterwarnings('ignore')

# Настройка отображения графиков
plt.style.use('default')
sns.set_palette("Blues")

# Загрузка данных
df = pd.read_csv('/content/drive/MyDrive/train.csv')

# Предварительный анализ данных
print("Размер датасета:", df.shape)
print("\nПервые 5 строк:")
print(df.head())

print("\nИнформация о данных:")
print(df.info())

# Обработка пропущенных значений
print("\nПропущенные значения:")
print(df.isnull().sum())

# Кодирование категориальных признаков
categorical_columns = df.select_dtypes(include=['object']).columns
print(f"\nКатегориальные признаки: {list(categorical_columns)}")

le = LabelEncoder()
for col in categorical_columns:
    df[col] = le.fit_transform(df[col].astype(str))

# Разделение на признаки и целевую переменную
X = df.drop('Price', axis=1)
y = df['Price']

# Разделение на train/test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

```
print(f"\nРазмеры выборок:")
print(f"Обучающая: X_train{X_train.shape}, y_train{y_train.shape}")
print(f"Тестовая: X_test{X_test.shape}, y_test{y_test.shape}")

print("\nОБУЧЕНИЕ БАЗОВЫХ МОДЕЛЕЙ")
print("=" * 50)

models = {
    'LinearRegression': LinearRegression(),
    'Lasso': Lasso(random_state=42),
    'ElasticNet': ElasticNet(random_state=42)
}

# Обучение моделей
trained_models = {}
for name, model in models.items():
    print(f"Обучение {name}...")
    model.fit(X_train, y_train)
    trained_models[name] = model

print("\nОЦЕНКА КАЧЕСТВА МОДЕЛЕЙ НА ТЕСТОВОЙ ВЫБОРКЕ")
print("=" * 50)

def evaluate_model(model, X_test, y_test, model_name):
    """Функция для оценки модели"""
    y_pred = model.predict(X_test)

    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print(f"\n{model_name}:")
    print(f"MSE: {mse:.2f}")
    print(f"RMSE: {rmse:.2f}")
    print(f"MAE: {mae:.2f}")
    print(f"R²: {r2:.4f}")

    return y_pred, {'MSE': mse, 'RMSE': rmse, 'MAE': mae, 'R²': r2}
# Оценка всех моделей
results = {}
predictions = {}

# График предсказанные vs фактические значения
plt.figure(figsize=(18, 5))

for i, (name, model) in enumerate(trained_models.items(), 1):
    y_pred, metrics = evaluate_model(model, X_test, y_test, name)
    predictions[name] = y_pred
    results[name] = metrics
```

```
    # График предсказанные vs фактические
    plt.subplot(1, 3, i)
    plt.scatter(y_test, y_pred, alpha=0.6, s=50)
    plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'r--', lw=2)
    plt.xlabel('Фактические значения')
    plt.ylabel('Предсказанные значения')
    plt.title(f'{name}\nR² = {metrics["R²"]:.4f}')
    plt.grid(alpha=0.3)

plt.tight_layout()
plt.show()

# Сравнение коэффициентов моделей
plt.figure(figsize=(18, 6))

# LinearRegression коэффициенты
plt.subplot(1, 3, 1)
coef_lr = trained_models['LinearRegression'].coef_
plt.barh(range(len(coef_lr)), coef_lr)
plt.title('LinearRegression - Коэффициенты')
plt.xlabel('Значение коэффициента')
plt.grid(axis='x', alpha=0.3)

# Lasso коэффициенты
plt.subplot(1, 3, 2)
coef_lasso = trained_models['Lasso'].coef_
plt.barh(range(len(coef_lasso)), coef_lasso)
plt.title('Lasso - Коэффициенты')
plt.xlabel('Значение коэффициента')
plt.grid(axis='x', alpha=0.3)

# ElasticNet коэффициенты
plt.subplot(1, 3, 3)
coef_elastic = trained_models['ElasticNet'].coef_
plt.barh(range(len(coef_elastic)), coef_elastic)
plt.title('ElasticNet - Коэффициенты')
plt.xlabel('Значение коэффициента')
plt.grid(axis='x', alpha=0.3)

plt.tight_layout()
plt.show()

# Сравнение метрик моделей
metrics_df = pd.DataFrame(results).T
print("\nСРАВНЕНИЕ МЕТРИК ВСЕХ МОДЕЛЕЙ:")
print(metrics_df.round(4))

# Визуализация сравнения метрик
fig, axes = plt.subplots(2, 2, figsize=(15, 10))
axes = axes.ravel()
```

```
for i, metric in enumerate(['R²', 'MSE', 'RMSE', 'MAE']):
    values = [results[model][metric] for model in results.keys()]
    bars = axes[i].bar(results.keys(), values, color=['skyblue',
'lightcoral', 'lightgreen'])
    axes[i].set_title(f'Сравнение {metric}')
    axes[i].set_ylabel(metric)

    # Добавляем значения на столбцы
    for bar, value in zip(bars, values):
        axes[i].text(bar.get_x() + bar.get_width()/2, bar.get_height(),
                     f'{value:.4f}', ha='center', va='bottom')

plt.tight_layout()
plt.show()

print("\nПОДБОР ГИПЕРПАРАМЕТРОВ И КРОСС-ВАЛИДАЦИЯ")
print("=" * 50)

# Создание пайплайнов с масштабированием
pipelines = {
    'Lasso': Pipeline([
        ('scaler', StandardScaler()),
        ('model', Lasso(random_state=42))
    ]),
    'ElasticNet': Pipeline([
        ('scaler', StandardScaler()),
        ('model', ElasticNet(random_state=42))
    ])
}# Параметры для GridSearch
param_grid = {
    'Lasso': {
        'model__alpha': [0.001, 0.01, 0.1, 1, 10, 100],
        'model__max_iter': [1000, 5000]
    },
    'ElasticNet': {
        'model__alpha': [0.001, 0.01, 0.1, 1, 10],
        'model__l1_ratio': [0.1, 0.3, 0.5, 0.7, 0.9],
        'model__max_iter': [1000, 5000]
    }
}

# Подбор гиперпараметров
best_models = {}
cv_results = {}

for name in ['Lasso', 'ElasticNet']:
    print(f"\nПодбор гиперпараметров для {name}...")

    grid_search = GridSearchCV(
        pipelines[name],
```

```
        param_grid[name],
        cv=5,
        scoring='r2',
        n_jobs=-1,
        verbose=1
    )

    grid_search.fit(X_train, y_train)
    best_models[name] = grid_search.best_estimator_
    cv_results[name] = grid_search.best_params_

    print(f"Лучшие параметры для {name}: {grid_search.best_params_}")
    print(f"Лучший R² на кросс-валидации: {grid_search.best_score_:.4f}")

# Кросс-валидация для всех моделей
print("\nРЕЗУЛЬТАТЫ КРОСС-ВАЛИДАЦИИ (5-fold):")
print("-" * 40)

# Добавляем LinearRegression в пайплайн для честного сравнения
linear_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('model', LinearRegression())
])

models_cv = {
    'LinearRegression': linear_pipeline,
    'Lasso': best_models['Lasso'],
    'ElasticNet': best_models['ElasticNet']
}

cv_scores = {}
for name, model in models_cv.items():
    scores = cross_val_score(model, X_train, y_train, cv=5, scoring='r2')
    cv_scores[name] = scores
    print(f"{name}: R² = {scores.mean():.4f} (+/- {scores.std() *
2:.4f})")

# Визуализация результатов кросс-валидации
plt.figure(figsize=(10, 6))
cv_data = []
for name, scores in cv_scores.items():
    for score in scores:
        cv_data.append({'Model': name, 'R²': score})

cv_df = pd.DataFrame(cv_data)
sns.boxplot(data=cv_df, x='Model', y='R²')
plt.title('Распределение R² по фолдам кросс-валидации')
plt.grid(axis='y', alpha=0.3)
plt.show()

# Сравнение моделей после подбора гиперпараметров
```

```
print("\nСРАВНЕНИЕ МОДЕЛЕЙ ПОСЛЕ ПОДБОРА ГИПЕРПАРАМЕТРОВ:")
print("=" * 50)

final_results = {}

for name, model in best_models.items():
    y_pred = model.predict(X_test)

    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    final_results[name] = {
        'MSE': mse, 'RMSE': rmse, 'MAE': mae, 'R²': r2
    }

    print(f"\n{name} (с подбором гиперпараметров):")
    print(f"R²: {r2:.4f}")
    print(f"RMSE: {rmse:.2f}")

# Визуализация сравнения моделей до и после подбора
models_comparison = pd.DataFrame({
    'Без подбора': [results['Lasso']['R²'], results['ElasticNet']['R²']],
    'С подбором': [final_results['Lasso']['R²'],
final_results['ElasticNet']['R²']]
}, index=['Lasso', 'ElasticNet'])

plt.figure(figsize=(10, 6))
models_comparison.plot(kind='bar', figsize=(10, 6))
plt.title('Сравнение качества моделей до и после подбора гиперпараметров')
plt.ylabel('R² Score')
plt.xticks(rotation=0)
plt.grid(axis='y', alpha=0.3)
plt.tight_layout()
plt.show()

print("\n1. Оптимальные гиперпараметры для Lasso и ElasticNet?")
for name in ['Lasso', 'ElasticNet']:
    print(f"   {name}: {cv_results[name]}")

print("\n2. Улучшение качества после подбора гиперпараметров")
for name in ['Lasso', 'ElasticNet']:
    improvement = final_results[name]['R²'] - results[name]['R²']
    percent_improvement = (improvement / results[name]['R²']) * 100
    print(f"   {name}: R² улучшился на {improvement:.4f} "
          f"({results[name]['R²']:.4f} ->
{final_results[name]['R²']:.4f})")
    print(f"   Относительное улучшение: {percent_improvement:.2f}%")
```