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-216
Русецкий А.С.
ФИО студента

Работу проверил:
Преподаватель
должность преподавателя
Сороковых Д.А.
ФИО преподавателя

Новосибирск 2025 г.

Введение

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

Задание

1. Предобработка данных
2. Обучите три базовые модели LinearRegression, Lasso и ElasticNet.
3. Оценка качества моделей на тестовой выборке
4. Подбор гиперпараметров и кросс-валидация

Основная часть

1. Подготовка данных

```
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, cross_val_score, GridSearchCV
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
```

```
# 1.1. Загрузка данных
df = pd.read_csv('Walmart_Sales.csv')
print("Размер датасета:", df.shape)
print("\nПервые 5 строк:")
df.head()
```

```
Размер датасета: (6435, 8)

Первые 5 строк:
```

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	8.106
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	8.106
2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	8.106
3	1	26-02-2010	1409727.59	0	46.63	2.561	211.319643	8.106
4	1	05-03-2010	1554806.68	0	46.50	2.625	211.350143	8.106

```python
# 1.2. Предобработка
print("Информация о датасете:")
print(df.info())
print("\nСтатистическое описание:")
print(df.describe())
```

```
Информация о датасете:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Store         6435 non-null   int64
 1   Date          6435 non-null   object
 2   Weekly_Sales  6435 non-null   float64
 3   Holiday_Flag  6435 non-null   int64
 4   Temperature   6435 non-null   float64
 5   Fuel_Price    6435 non-null   float64
 6   CPI           6435 non-null   float64
 7   Unemployment  6435 non-null   float64
dtypes: float64(5), int64(2), object(1)
memory usage: 402.3+ KB
None

Статистическое описание:
              Store  Weekly_Sales  Holiday_Flag  Temperature   Fuel_Price  \
count  6435.000000  6.435000e+03   6435.000000  6435.000000  6435.000000
mean     23.000000  1.046965e+06      0.069930    60.663782     3.358607
std      12.988182  5.643666e+05      0.255049    18.444933     0.459020
min       1.000000  2.099862e+05      0.000000    -2.060000     2.472000
25%      12.000000  5.533501e+05      0.000000    47.460000     2.933000
50%      23.000000  9.607460e+05      0.000000    62.670000     3.445000
75%      34.000000  1.420159e+06      0.000000    74.940000     3.735000
max      45.000000  3.818686e+06      1.000000   100.140000     4.468000

                CPI  Unemployment
count  6435.000000   6435.000000
mean    171.578394      7.999151
std      39.356712      1.875885
min     126.064000      3.879000
25%     131.735000      6.891000
50%     182.616521      7.874000
75%     212.743293      8.622000
max     227.232807     14.313000
```

```python
print("Пропущенные значения:")
print(df.isnull().sum())

if df.isnull().sum().sum() > 0:
    numeric_columns = df.select_dtypes(include=[np.number]).columns
    df[numeric_columns] = df[numeric_columns].fillna(df[numeric_columns].median())

    categorical_columns = df.select_dtypes(include=['object']).columns
    for col in categorical_columns:
        df[col] = df[col].fillna(df[col].mode()[0] if not df[col].mode().empty else 'Unknown')
```

```
Пропущенные значения:
Store           0
Date            0
Weekly_Sales    0
Holiday_Flag    0
Temperature     0
Fuel_Price      0
CPI             0
Unemployment    0
dtype: int64
```

```python
label_encoders = {}
categorical_columns = df.select_dtypes(include=['object']).columns

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

print(df.head())
```

```
   Store  Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
0      1    19    1643690.90             0        42.31       2.572
1      1    52    1641957.44             1        38.51       2.548
2      1    85    1611968.17             0        39.93       2.514
3      1   118    1409727.59             0        46.63       2.561
4      1    20    1554806.68             0        46.50       2.625

          CPI  Unemployment
0  211.096358         8.106
1  211.242170         8.106
2  211.289143         8.106
3  211.319643         8.106
4  211.350143         8.106
```

```python
if 'Weekly_Sales' in df.columns:
    target_col = 'Weekly_Sales'

print(f"Целевая переменная: {target_col}")

X = df.drop(columns=[target_col])
y = df[target_col]

print(f"Размеры: X {X.shape}, y {y.shape}")
```

```
Целевая переменная: Weekly_Sales
Размеры: X (6435, 7), y (6435,)
```

```python
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

print(f"Обучающая выборка: {X_train.shape}")
print(f"Тестовая выборка: {X_test.shape}")
```

```
Обучающая выборка: (5148, 7)
Тестовая выборка: (1287, 7)
```

2. Базовые модели

```python
# 2.1. Обучение базовых моделей
models = {
    'LinearRegression': LinearRegression(),
    'Lasso': Lasso(random_state=42),
    'ElasticNet': ElasticNet(random_state=42)
}

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

```
Модель LinearRegression обучена
Модель Lasso обучена
Модель ElasticNet обучена
```

Lasso - Проще говоря, лассо-регрессия стремится уменьшить число параметров путем зануления весов для неинформативных и избыточных признаков, что на выходе даст разреженную модель (с небольшим числом ненулевых весов признаков)

ElasticNet - представляет собой комбинацию L1 и L2-регуляризаций через отношение их смеси r, что может принести особую пользу в ситуациях, когда в данных необходимо одновременно выполнять отбор признаков и бороться с мультиколлинеарностью.

В L1-регуляризации используется сумма абсолютных значений весовых значений, а в L2-регуляризации — сумма квадратов весовых значений.

3. Оценка качества моделей на тестовой выборке

```python
# 3.1. Оценка качества моделей
def evaluate_model(model, X_test, y_test, model_name):
    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)

    metrics = {
        'MSE': mse,
        'RMSE': rmse,
        'MAE': mae,
        'R2': r2
    }

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

    return metrics, y_pred

# Оценка всех моделей
results = {}
predictions = {}

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

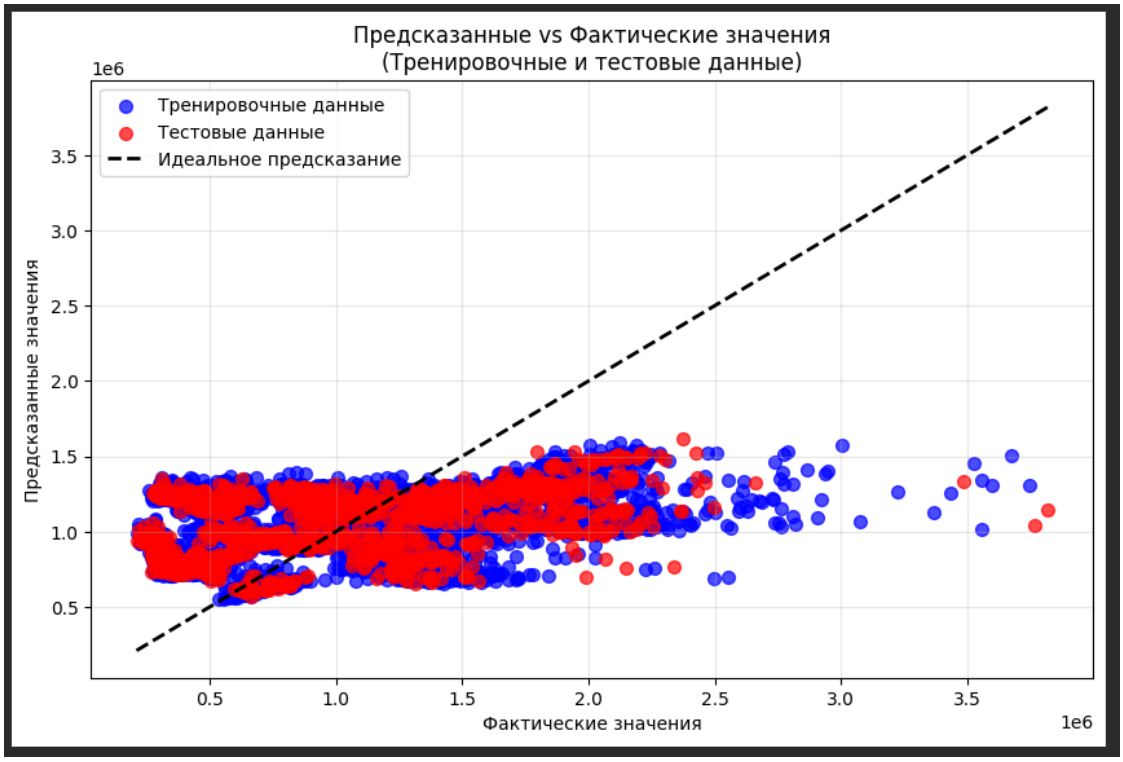
```
LinearRegression:
MSE: 274749075432.0847
RMSE: 524165.1223
MAE: 433488.1751
R²: 0.1472

Lasso:
MSE: 274748866716.5457
RMSE: 524164.9232
MAE: 433487.9401
R²: 0.1472

ElasticNet:
MSE: 274611022994.2295
RMSE: 524033.4178
MAE: 433333.3351
R²: 0.1476
```

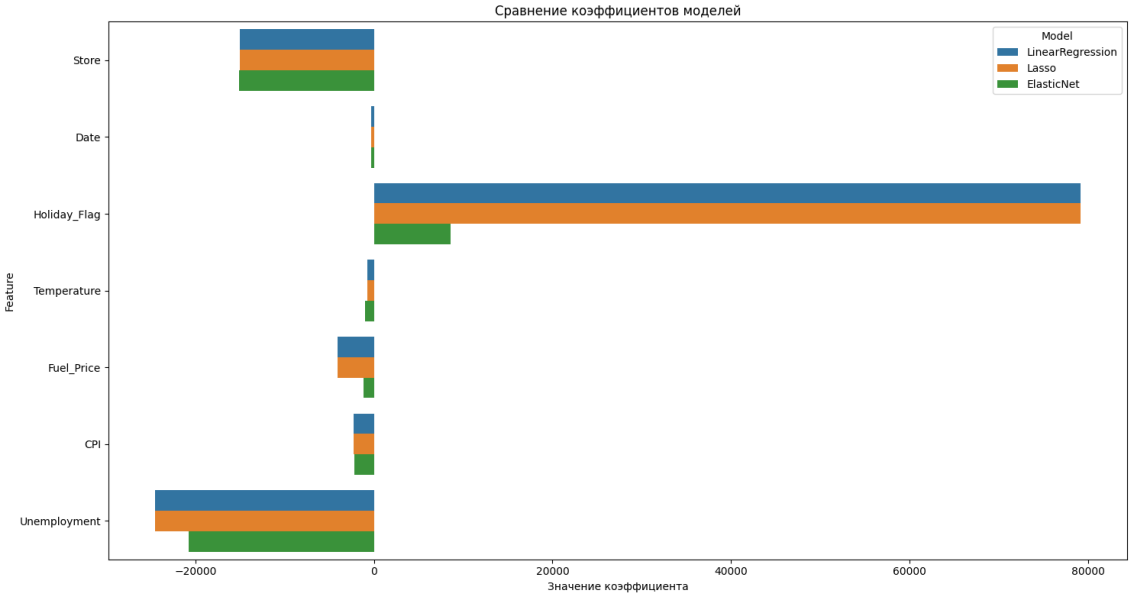
MSE - средневадратичная ошибка (разница между предсказанием и реальным значением)

RMSE - корень MSE

MAE - средняя абсолютная ошибка (берём среднее из квадрата разницы между предсказанным и реальным)

R^2 - коэффициент детерминации 14% очень мало (не лучше чем предсказание средним значением)





4. Подбор гиперпараметров и кросс-валидация

```python
# 4.1. Подбор гиперпараметров для Lasso и ElasticNet
param_grid = {
    'Lasso': {
        'alpha': [0.001, 0.01, 0.1, 1.0, 10.0],
        'max_iter': [1000, 5000]
    },
    'ElasticNet': {
        'alpha': [0.001, 0.01, 0.1, 1.0, 10.0],
        'l1_ratio': [0.1, 0.3, 0.5, 0.7, 0.9],
        'max_iter': [1000, 5000]
    }
}

tuned_models = {}

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

    if model_name == 'Lasso':
        model = Lasso(random_state=42)
    else:
        model = ElasticNet(random_state=42)

    grid_search = GridSearchCV(
        model,
        param_grid[model_name],
        cv=5,
        scoring='neg_mean_squared_error',
        n_jobs=-1
    )

    grid_search.fit(X_train, y_train)
    tuned_models[model_name] = grid_search.best_estimator_

    print(f"Лучшие параметры для {model_name}: {grid_search.best_params_}")
    print(f"Лучший MSE: {grid_search.best_score_:.4f}")
```

```
Подбор гиперпараметров для Lasso
Лучшие параметры для Lasso: {'alpha': 10.0, 'max_iter': 1000}
Лучший MSE: -273806983269.4623

Подбор гиперпараметров для ElasticNet
Лучшие параметры для ElasticNet: {'alpha': 0.1, 'l1_ratio': 0.7, 'max_iter': 1000}
Лучший MSE: -273758330827.1323
```

```python
# 4.2. Кросс-валидация для всех моделей
models_cv = {
    'LinearRegression': LinearRegression(),
    'Lasso': tuned_models.get('Lasso', Lasso(random_state=42)),
    'ElasticNet': tuned_models.get('ElasticNet', ElasticNet(random_state=42)),
    'Lasso_tuned': tuned_models.get('Lasso'),
    'ElasticNet_tuned': tuned_models.get('ElasticNet')
}

models_cv = {k: v for k, v in models_cv.items() if v is not None}

cv_results = {}

for name, model in models_cv.items():
    mse_scores = -cross_val_score(model, X_train, y_train,
                                  cv=5, scoring='neg_mean_squared_error')

    r2_scores = cross_val_score(model, X_train, y_train,
                                cv=5, scoring='r2')

    cv_results[name] = {
        'MSE_mean': mse_scores.mean(),
        'MSE_std': mse_scores.std(),
        'R2_mean': r2_scores.mean(),
        'R2_std': r2_scores.std()
    }

    print(f"\n{name} - Кросс-валидация:")
    print(f"MSE: {mse_scores.mean():.4f} (+/- {mse_scores.std() * 2:.4f})")
    print(f"R²: {r2_scores.mean():.4f} (+/- {r2_scores.std() * 2:.4f})")
```

```
LinearRegression - Кросс-валидация:
MSE: 273807565263.2724 (+/- 19702552562.4546)
R²: 0.1373 (+/- 0.0286)

Lasso - Кросс-валидация:
MSE: 273806983269.4623 (+/- 19704887621.9011)
R²: 0.1373 (+/- 0.0286)

ElasticNet - Кросс-валидация:
MSE: 273758330827.1323 (+/- 20108381202.5890)
R²: 0.1375 (+/- 0.0272)

Lasso_tuned - Кросс-валидация:
MSE: 273806983269.4623 (+/- 19704887621.9011)
R²: 0.1373 (+/- 0.0286)

ElasticNet_tuned - Кросс-валидация:
MSE: 273758330827.1323 (+/- 20108381202.5890)
R²: 0.1375 (+/- 0.0272)
```

```python
# Сравнение качества до и после подбора гиперпараметров
comparison_df = pd.DataFrame(cv_results).T
print("Сравнение результатов кросс-валидации:")
print(comparison_df)

mse_means = [cv_results[name]['MSE_mean'] for name in cv_results.keys()]
mse_stds = [cv_results[name]['MSE_std'] for name in cv_results.keys()]

r2_means = [cv_results[name]['R2_mean'] for name in cv_results.keys()]
r2_stds = [cv_results[name]['R2_std'] for name in cv_results.keys()]
```

```
Сравнение результатов кросс-валидации:
                     MSE_mean       MSE_std   R2_mean    R2_std
LinearRegression  2.738076e+11  9.851276e+09  0.137345  0.014294
Lasso             2.738070e+11  9.852444e+09  0.137347  0.014289
ElasticNet        2.737583e+11  1.005419e+10  0.137534  0.013601
Lasso_tuned       2.738070e+11  9.852444e+09  0.137347  0.014289
ElasticNet_tuned  2.737583e+11  1.005419e+10  0.137534  0.013601
```

Заключение

В ходе работы были протестированы несколько моделей — LinearRegression, Lasso и ElasticNet.

По рассчитанным метрикам (MSE, RMSE, MAE и R^2) можно сказать, что моделям удалось добиться среднего качества прогнозирования.

В целом модели работают, но их точность нельзя назвать высокой.

Ссылка на google colab:
https://colab.research.google.com/drive/1PTdUpgyOg8xPAsKkyl1rO9u0o4VrUbDM?usp=sharing