УДК 004

Разведочный анализ данных о прогнозировании энергопотребления

Матвеева Алёна Сергеевна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
Целью исследования является построение нескольких моделей регрессии по данным прогнозирования энергопотребления. Для реализации использовалась свободно распространяемая платформа Google Colab. Данное исследование может быть использовано методическим пособием в учебной деятельности.
Ключевые слова: Python, модель, таблица.

Exploratory analysis of energy consumption forecasting data

Matveeva Alyona Sergeevna
Sholom-Aleichem Priamursky State University
Student

Abstract
The aim of the study is to build several regression models based on energy consumption forecasting data. The freely distributed Google Scholar platform was used for implementation. This study can be used as a methodological guide in educational activities.
Keywords: Python, model, table.

1 Введение

1.1 Актуальность

Актуальностью данной темы заключается в широком использовании и представлении информации. Путем построения нескольких моделей регрессии можно сравнить их производительность и выбрать наиболее точную модель для предсказания значений. Это может привести к улучшению точности предсказаний и более надежным результатам. А также построение нескольких моделей позволяет обнаружить выбросы и аномалии в данных. Разные модели могут реагировать по-разному на выбросы, и анализ результатов нескольких моделей может помочь выявить аномалии и принять соответствующие меры, такие как исключение выбросов из анализа или проведение дополнительных исследований.

1.2 Обзор исследований

В статье Г.С. Осипов и Н.С. Вашакидзе привели пример синтеза нейросетевой модели регрессии-авторегрессии для решения задачи прогнозирования [2]. Н.Б. Паклин, В.И. Орешков в статье приведен обзор бизнес-аналитике [3]. А также в статье О.В. Кудринской рассматриваются

возможности языка Python и популярные библиотеки, позволяющие визуализировать данные [4]. Е.В. Вишнякова, Е.В.Иванова, С.М.Камалов, Ю.А.Колодяжная, Л.Ф. Хамидуллина в своей статье проводят анализ нечеткой линейной регрессия в задачах оценки [5]. В англоязычной статье R. Aurachman рассматривает визуализацию данных с использованием программирования python [6].

1.3 Цель исследования

Целью исследования является построение нескольких моделей регрессии по данным прогнозирования энергопотребления.

2 Материалы и методы

В данном исследовании используется платформа Google Colab для написания кода на языке программировании Python. Материалы с данными можно скачать по ссылке [1].

Для построения и оценки моделей использованы следующие методы:

Линейная регрессия: Простая модель, позволяющая оценить линейные отношения между признаками и целевой переменной.

Случайный лес: Ансамблевая модель, объединяющая несколько деревьев решений для повышения обобщающей способности.

XGBoost: Эффективный алгоритм градиентного бустинга, придающий больший вес ошибкам предыдущих моделей.

Дерево решений: Нелинейная модель, способная улавливать сложные зависимости в данных.

3 Результаты

Для данной статье подключим библиотеки и создадим модели «Линейная регрессия» и «Случайный лес», а также объект StandardScaler для предварительной обработки признаков:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
forest = RandomForestRegressor()
lr = LinearRegression()
scaler = StandardScaler()
```

Загружаем файл для работы с данными и отображаем таблицу первые 5 строк (рис. 1):

```
df = pd.read_csv('Energy_consumption.csv')
df.head()
```

	Timestamp	Temperature	Humidity	SquareFootage	Occupancy	HVACUsage	LightingUsage	RenewableEnergy	DayOfWeek	Holiday	EnergyConsumption
0	2022-01-01 00:00:00	25.139433	43.431581	1565.693999	5	On	Off	2.774699	Monday	No	75.364373
1	2022-01-01 01:00:00	27.731651	54.225919	1411.064918	1	On	On	21.831384	Saturday	No	83.401855
2	2022-01-01 02:00:00	28.704277	58.907658	1755.715009	2	Off	Off	6.764672	Sunday	No	78.270888
3	2022-01-01 03:00:00	20.080469	50.371637	1452.316318	1	Off	On	8.623447	Wednesday	No	56.519850
4	2022-01-01 04:00:00	23.097359	51.401421	1094.130359	9	On	Off	3.071969	Friday	No	70.811732

Рисунок 1 – Отображение таблицы «df»

Выводим информация о структуре данных в «df» и основные статистические показатели (рис. 2):

```
df.info()
output
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Timestamp         1000 non-null   object
 1   Temperature       1000 non-null   float64
 2   Humidity          1000 non-null   float64
 3   SquareFootage     1000 non-null   float64
 4   Occupancy         1000 non-null   int64
 5   HVACUsage         1000 non-null   object
 6   LightingUsage     1000 non-null   object
 7   RenewableEnergy   1000 non-null   float64
 8   DayOfWeek         1000 non-null   object
 9   Holiday           1000 non-null   object
 10  EnergyConsumption 1000 non-null   float64
dtypes: float64(5), int64(1), object(5)
memory usage: 86.1+ KB
df.describe()
```

	Temperature	Humidity	SquareFootage	Occupancy	RenewableEnergy	EnergyConsumption
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	24.982026	45.395412	1500.052488	4.581000	15.132813	77.055873
std	2.836850	8.518905	288.418873	2.865598	8.745917	8.144112
min	20.007565	30.015975	1000.512661	0.000000	0.006642	53.263278
25%	22.645070	38.297722	1247.108548	2.000000	7.628385	71.544690
50%	24.751637	45.972116	1507.967426	5.000000	15.072296	76.943696
75%	27.418174	52.420066	1740.340165	7.000000	22.884064	82.921742
max	29.998671	59.969085	1999.982252	9.000000	29.965327	99.201120

Рисунок 2 – Основные статистические показатели

Создаем график с ящиками для визуализации распределения данных по различным признакам (рис. 3):

```
      fig, axes = plt.subplots(2, 3, figsize=(13, 7))
      sns.boxplot(x=df['Temperature'], ax=axes[0][0]) #Температура
      sns.boxplot(x=df['Humidity'], ax=axes[0][1]) #Влажность
      sns.boxplot(x=df['SquareFootage'], ax=axes[0][2]) #Площадь помещения
      sns.boxplot(x=df['Occupancy'], ax=axes[1][0]) #Загруженность
      sns.boxplot(x=df['RenewableEnergy'],    ax=axes[1][1])    #Возобновляемая
энергия
      sns.boxplot(x=df['EnergyConsumption'],    ax=axes[1][2])    #Потребление
энергии
```

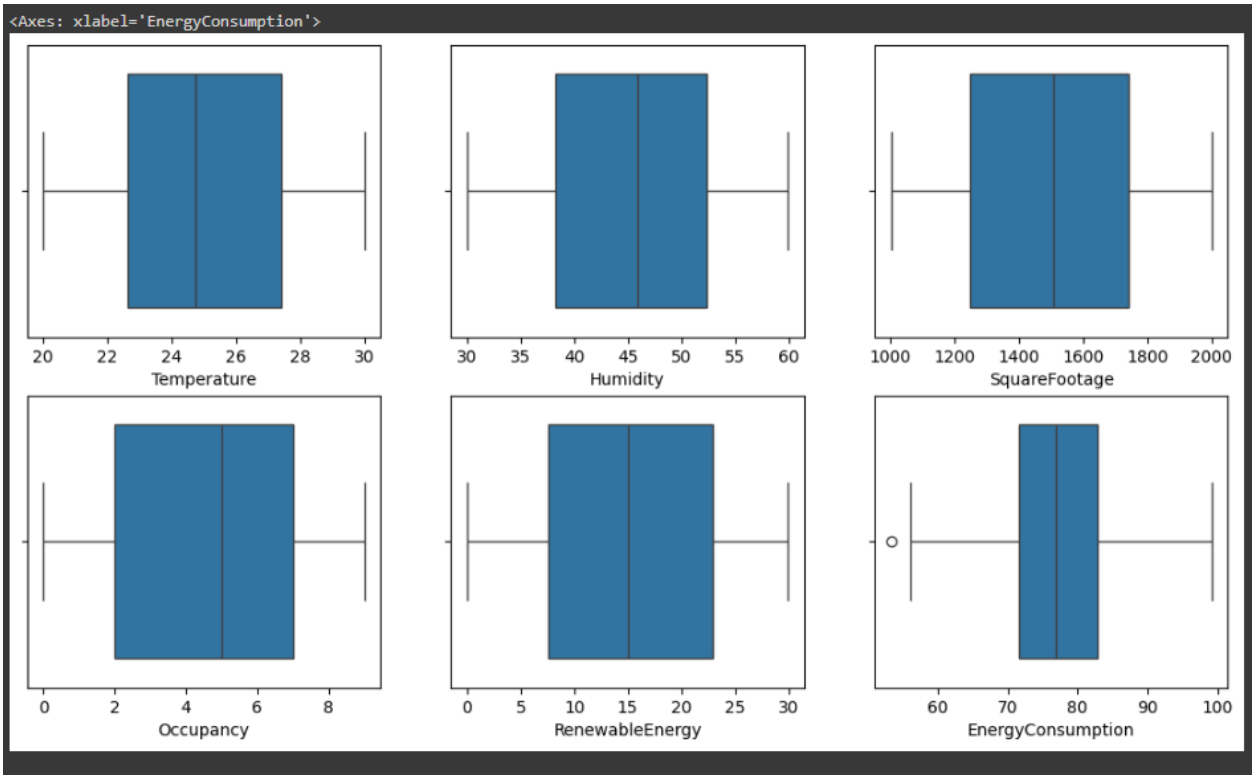

Рисунок 3 – График с ящиками

Определяем функцию total_outliers для подсчета выбросов в столбце EnergyConsumption:

```
      def total_outliers(column):
          column_df = pd.DataFrame(column)
          q1 = column.quantile(0.25)
          q3 = column.quantile(0.75)
          IQR = q3 - q1
                  outliers1     =     column_df[(column_df<(q1-1.5*IQR))      |
(column_df>(q3+1.5*IQR))]
          hasil = outliers1.count()
          return hasil
      total_outliers(df['EnergyConsumption'])
```

```
EnergyConsumption  1
dtype: int64
```

Выполняем удаление строк с минимальным значением «EnergyConsumption» и столбца «Timestamp» из «df»:

```
      df.drop(df[df['EnergyConsumption']                                    ==
df['EnergyConsumption'].min()].index, axis=0, inplace=True)
      df.drop(['Timestamp'], axis=1, inplace=True)
```

Производится преобразование категориальных признаков в числовые с помощью метода get_dummies:

```
df = pd.get_dummies(df).astype(int)
```

Создаем модель для предсказания значения «EnergyConsumption» на основе остальных признаков. Данные разделяем на обучающую и тестовую выборки, масштабируем с использованием «StandardScaler»:

```
X = df.drop('EnergyConsumption', axis=1)
y = df['EnergyConsumption']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
X_train_s = scaler.fit_transform(X_train)
X_test_s = scaler.fit_transform(X_test)
lr.fit(X_train, y_train)
```

Обучаем модель LinearRegression и вычисляем коэффициент детерминации на тестовой выборке:

```
lr_score = lr.score(X_test, y_test)
print(lr_score)
output
0.5495966243016259
```

Вычисляем среднеквадратичную ошибку для модели LinearRegression:

```
from sklearn.metrics import mean_squared_error
y_pred = lr.predict(X_test)
lr_mse = mean_squared_error(y_test, y_pred)
print(lr_mse)
output
26.590193849079498
```

Обучаем модель RandomForestRegressor и вычисляем коэффициент детерминации на тестовой выборке:

```
forest.fit(X_train, y_train)
forest_score = forest.score(X_test, y_test)
print(forest_score)
output
0.5145258687860371
```

Вычисляем среднеквадратичную ошибку для модели RandomForestRegressor:

```
yf_pred = forest.predict(X_test)
forest_mse = mean_squared_error(y_test, yf_pred)
print(forest mse)
output
28.660645000000002
```

Производим поиск оптимальных параметров модели RandomForestRegressor с использованием GridSearchCV:

```
from sklearn.model_selection import GridSearchCV
param_grid = {
    'n_estimators':[600],
    'max_depth':[12],
    'min_samples_split':[6],
    'max_features':['sqrt'] }
grid_search = GridSearchCV(forest, param_grid, cv=5,
                           scoring='neg_mean_squared_error',
                           return_train_score=True)
grid_search.fit(X_train, y_train)
```

Выводим лучшую модель и ее коэффициент детерминации на тестовой выборке:

```
grid_search.best_estimator_
grid_search.best_estimator_.score(X_test, y_test)
output
```

```
0.5622856207089937
```

Вычисляем среднеквадратичную ошибку для лучшей модели RandomForestRegressor:

```
ygs_pred = grid_search.predict(X_test)
print(mean_squared_error(y_test, ygs_pred))
output
25.841081181575568
```

Подключаем библиотеку для модели XGBoost:

```
import xgboost as xgb
```

Обучение модели градиентного бустинга XGBoost:

```
xgb_model = xgb.XGBRegressor()
xgb_model.fit(X_train_s, y_train)
```

Оценка модели на тестовой выборке:

```
score = xgb_model.score(X_test_s, y_test)
print("R^2 score:", score)
output
R^2 score: 0.17252540565829177
```

Вычисляем среднеквадратичную ошибку для лучшей модели XGBoost:

```
xgb_pred = xgb_model.predict(X_test)
xgb_mse = mean_squared_error(y_test, xgb_pred)
print(xgb_mse)
output
149.21349010925974
```

Визуализация важности признаков для данной модели (рис. 4):

```
fig, ax = plt.subplots(figsize=(10, 6))
xgb.plot_importance(xgb_model, ax=ax)
plt.show()
```

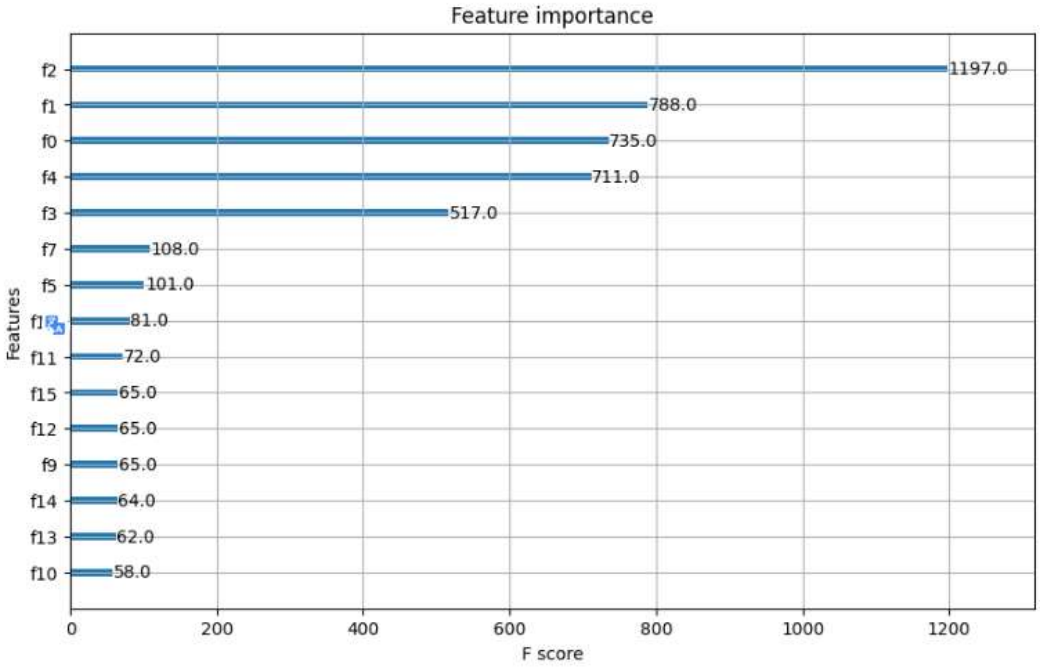

Рисунок 4 – Визуализация важности признаков модели XGBoost

Визуализация важности признаков, полученная с помощью функции xgb.plot_importance, показывает, насколько каждый признак внес вклад в прогнозирование целевой переменной (в данном случае - потребление энергии).

На графике признаки представлены по оси Y, а их важность (значение) - по оси X. Чем выше значение важности, тем больший вклад вносит соответствующий признак в предсказание модели.

В данном коде признаки были закодированы с помощью метода «One-Hot Encoding», поэтому каждый признак представлен в виде нескольких столбцов (бинарные переменные). Важность каждого признака рассчитывается как сумма важностей всех его бинарных переменных.

Визуализация важности признаков позволяет определить наиболее значимые признаки, которые оказывают наибольшее влияние на прогнозирование потребления энергии. Это может быть полезной информацией для понимания данных и оптимизации модели.

Подключение библиотеки для модели DecisionTreeRegressor:

```
from sklearn.tree import DecisionTreeRegressor
from sklearn import tree
```

Создание модели DecisionTreeRegressor:

```
dt = DecisionTreeRegressor()
# Обучение модели на тренировочных данных
dt.fit(X_train, y_train)
```

Оценка модели на тестовых данных:

```
score1 = dt.score(X_test, y_test)
print("R-squared score:", score1)
output
R-squared score: 0.01408283702935842
```

Вычисляем среднеквадратичную ошибку для модели DecisionTreeRegressor:

```
dt_pred = dt.predict(X_test)
dt_mse=mean_squared_error(y_test, dt_pred)
print(dt_mse)
output
58.205
```

Визуализация дерева решений (рис. 5):

```
plt.figure(figsize=(15, 10))
tree.plot_tree(dt, filled=True)
plt.show()
```

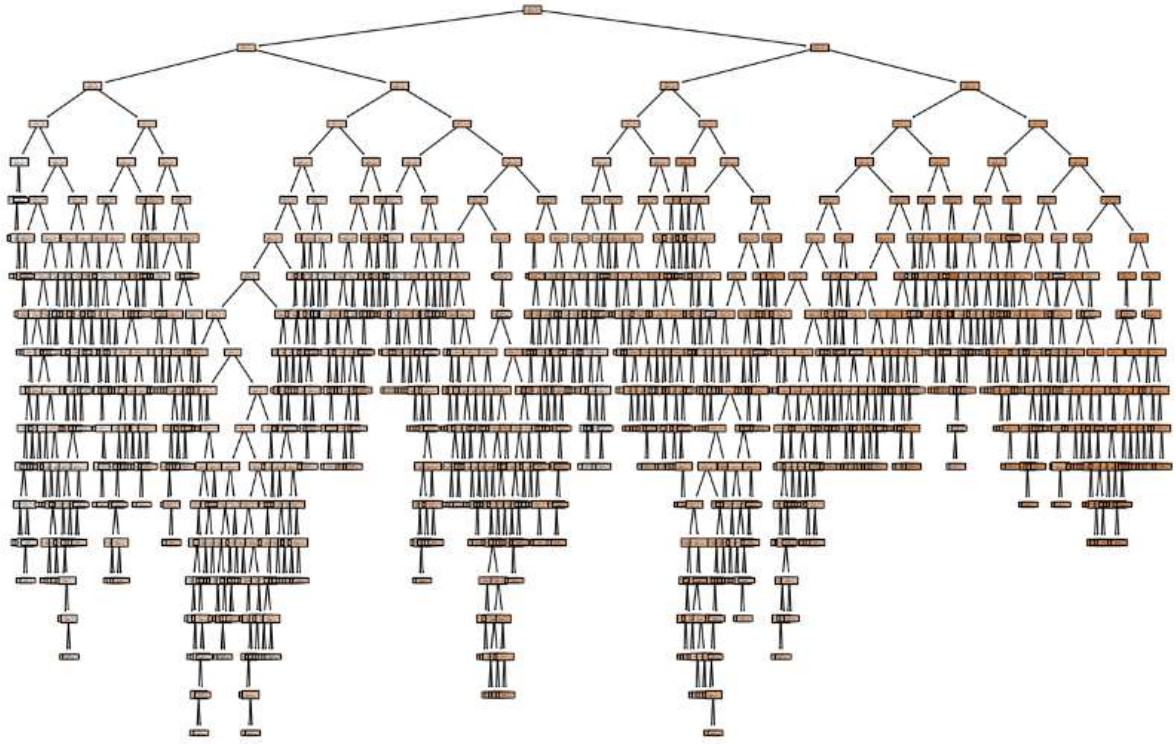

Рисунок 5 – Визуализация дерева решений

Визуализация дерева решений представляет собой графическое представление структуры дерева, где каждый узел представляет собой разделение данных на основе определенного признака, а каждое листовое (терминальное) значение представляет прогнозируемый результат или значение целевой переменной.

На визуализации дерева решений можно увидеть следующие элементы:

Узлы (ноды): Каждый узел представляет собой разделение данных на основе определенного признака. Узлы имеют условия, по которым данные разделяются на две или более ветви.

Ветви: Каждая ветвь представляет собой возможное значение признака, по которому происходит разделение данных.

Листовые (терминальные) узлы: Листовые узлы представляют собой конечные значения или прогнозы для целевой переменной. Они не имеют дальнейших ветвей или разделений. Визуализация дерева решений полезна для понимания того, как модель принимает решения на основе признаков и какие признаки наиболее важны для прогнозирования целевой переменной. Она также может помочь в идентификации переобучения или недообучения модели.

Создание таблицы для сравнения полученных данных:

```
      data = {
           'Модель': ['LinearRegression', 'RandomForestRegressor', 'XGBoost',
'DecisionTreeRegressor'],
          'Коэффициент детерминации': [lr_score, forest_score, score, score1],
          'Среднеквадратичная ошибка': [lr_mse, forest_mse, xgb_mse, dt_mse] }
      df1 = pd.DataFrame(data)
      # Вывод таблицы
```

```
print(df1)
output
                Модель  Коэффициент детерминации  Среднеквадратичная ошибка
0     LinearRegression                  0.549597                   26.590194
1  RandomForestRegressor                0.514526                   28.660645
2              XGBoost                  0.172525                  149.213490
3  DecisionTreeRegressor                0.014083                   58.205000
```

Исходя из таблицы, модель LinearRegression имеет наилучший коэффициент детерминации (0.549597), что говорит о лучшей предсказательной способности этой модели по сравнению с другими моделями. Кроме того, у LinearRegression также наименьшая среднеквадратичная ошибка (26.590194), что указывает на более точные предсказания этой модели.

4 Выводы

Таким образом, в данной статье был рассмотрен процесс построение нескольких моделей регрессии: «Линейная регрессия», «Случайный лес», «XGBoost» и «Дерево решений» по данным прогнозирования энергопотребления. Исходя из таблицы сравнений моделей, лучшей предсказательной способности модели была «Линейная регрессия». Данное исследование может быть использовано методическим пособием в учебной деятельности.

Библиографический список

1. Данные URL: https://drive.google.com/drive/folders/1B7yV-oYWqR28rfmHLVEF5x0IglSKnt2V?usp=sharing
2. Осипов Г.С., Вашакидзе Н.С. Построение нейросетевых моделей типа «регрессия-авторегрессия» на основе аналитической платформы Deductor // Постулат. 2017. №8. С. 17.
3. Паклин Н.Б., Орешков В.И. Бизнес-аналитика: от данных к знаниям. СПб.: Питер, 2013. 704 с.
4. Кудринская О.В. Визуализация данных с использованием возможностей языка python // Сборник научных статей ежегодной научно-практической конференции. 2021. С. 176-180.
5. Вишнякова Е.В., Иванова Е.В., Камалов С.М., Колодяжная Ю.А., Хамидуллина Л.Ф. Нечеткая линейная регрессия в задачах оценки//Научные записки молодых исследователей. 2015. № 5. С. 14–29.
6. Aurachman R. Visualization of google mobility data for provinces in Indonesia using seaborn python programming package //Journal of Physics: Conference Series. IOP Publishing, 2021. Т. 1833. №. 1. С. 012002.