

РАЗРАБОТКА МОДЕЛИ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ПРОГНОЗИРОВАНИЯ ОБЪЕМОВ ПРОДАЖ В РИТЕЙЛЕ НА ПРИМЕРЕ ФЕДЕРАЛЬНОЙ РОЗНОЧНОЙ СЕТИ

Подготовлено Виноградовой С.Р. под
руководством Семендяева Р.Ю.

24.05.2023



Agenda

- ▶ Выбранная проблема и ее актуальность
- ▶ ML-методы прогнозирования
- ▶ Практическое использование алгоритмов ML в прогнозировании объемов продаж в ритейле
- ▶ Выводы



Выбранная проблема и ее актуальность

Российский фуд-ритейл высококонкурентен! Чтобы удерживать лидирующие позиции, компании стремятся к высокотехнологичности!

- ▶ Крупные федеральные ритейлеры:



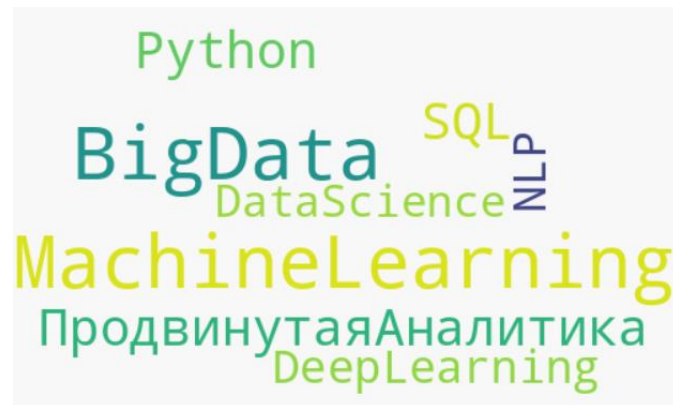
- ▶ Международные сети:



- ▶ Dark-stores, занимающиеся онлайн заказами и доставкой на дом:



- ▶ Региональные и локальные игроки



.... и др.

Выбранная проблема и ее актуальность

Задачи ритейла, которые могут быть решены с помощью инструментов Machine Learning (ML):

- ▶ Прогнозирование объемов продаж;
- ▶ Персонализация и рекомендации;
- ▶ Управление запасами;
- ▶ Ценообразование;
- ▶ Анализ и обработка данных;
- ▶ И другие.

Выгода для компании:

- ★ Оптимизация операционной деятельности;
- ★ Улучшение обслуживания покупателей;
- ★ Увеличение прибыльности;
- ★ Облегчение принятия стратегических решений;
- ★ Повышение конкурентоспособности!

ML-методы прогнозирования

Линейные модели

▶ $y = w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n$, где:

y – целевая (выходная) переменная, т.е. то значение, которое должно быть предсказано;

x_1, x_2, \dots, x_n - входные признаки;

$w_0, w_1, w_2, \dots, w_n$ - параметры модели, которые нужно определить.

- ▶ Линейная регрессия (LinearRegression),
- ▶ Гребневая регрессия (Ridge),
- ▶ Лассо регрессия (Lasso),
- ▶ Эластичная сеть (ElasticNet),
- ▶ Линейный метод опорных векторов (LinearSVR),
- ▶ Стохастический градиентный спуск (SGDRegressor).

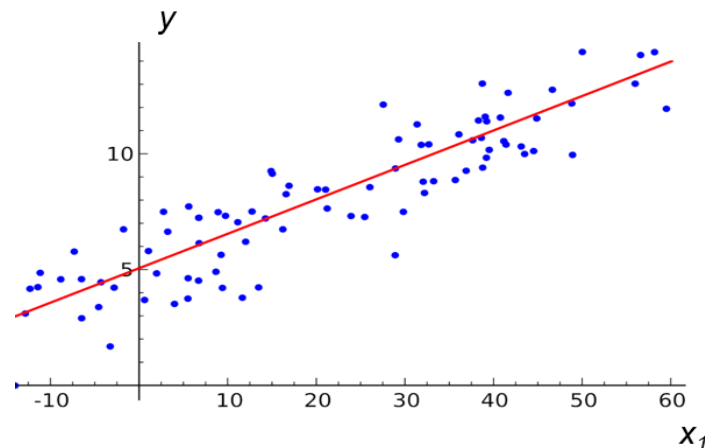


Рис.1 – Пример графика линейной регрессии

Пример работы бустинга

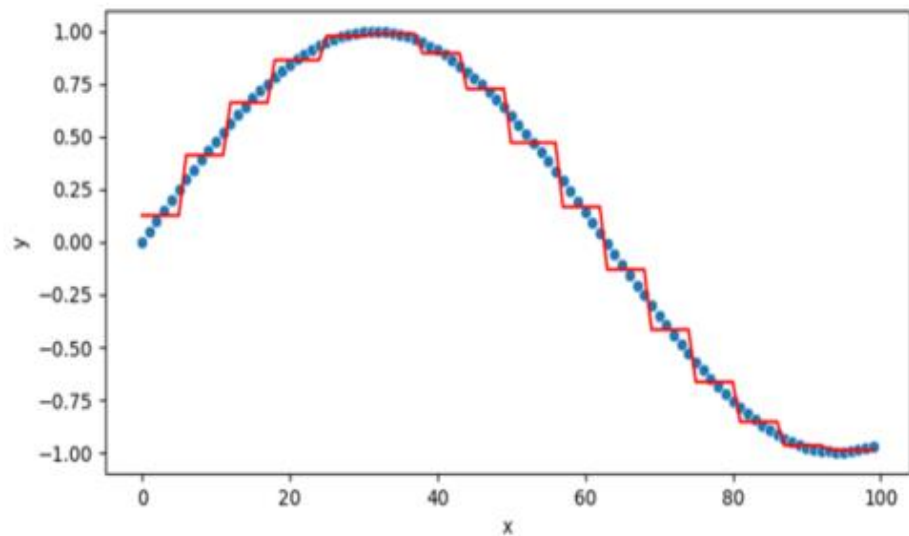


Рис.5 – Пример работы бустинга

Практическое использование алгоритмов ML в прогнозировании объемов продаж в ритейле

Предобработка и описательный анализ данных - Это два очень важных этапа, являющиеся неотъемлемой частью процесса построения моделей машинного обучения, так как гарантируют качество данных, а, следовательно, и достоверность результатов, и эффективность моделей машинного обучения.

```
df.shape
```

```
(829872, 7)
```

```
df.head()
```

	Дата	День недели	Код Региона	Признак Промо	Признак ЧТМ	Группа Закупок	РТО, руб. без НДС
0	20210501	6	CD06	Promo	PrLabel	P05	943526.34
1	20210501	6	CD02	Promo	PrLabel	P16	6886.58
2	20210501	6	CD06	Promo	Brand	P21	2901613.85
3	20210501	6	CD05	Promo	PrLabel	P31	716275.46
4	20210501	6	CD04	Promo	Brand	P35	111997.60

Исходные данные - таблица размерностью 829 872 x 7 (829 872 строки, 7 столбцов) с розничными продажами РТО, руб. без НДС (целевая переменная) в период с 01.05.2021 по 01.05.2023 с разбивкой по различным признакам

Практическое использование алгоритмов ML в прогнозировании объемов продаж в ритейле

Этапы предобработки и описательного анализа данных :

1. Определение типов переменных:

```
df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 829872 entries, 0 to 415615
Data columns (total 7 columns):
 #   Column                Non-Null Count  Dtype
---  -
 0   дата                  829872 non-null int64
 1   День_недели           829872 non-null int64
 2   Код_Региона           829872 non-null object
 3   Признак_промо         829872 non-null object
 4   Признак_ЧТМ           829872 non-null object
 5   Группа_Закупок        829872 non-null object
 6   РТО_руб._без_НДС      829872 non-null float64
dtypes: float64(1), int64(2), object(4)
```

Рис.6 – Типы переменных данных

```
df['Дата'] =
pd.to_datetime(df['Дата'].astype(str))
```

2. Ввод новых признаков:

```
#добавление признака 'Месяц'
df['Месяц'] = df['Дата'].dt.month
#добавление признака 'День'
df['День'] = df['Дата'].dt.day
#добавление признака 'Выходной'
def wd(day):
    if day > 4:
        return 1
    else:
        return 0

df['Выходной'] = df.apply(lambda x:
wd(x['День_недели']), axis=1)
#добавление признака 'Праздник'
ru_holidays = holidays.Russia(years=[2021, 2022, 2023])
df['Праздник'] = df['Дата'].isin(ru_holidays).astype(int)
```

Практическое использование алгоритмов ML в прогнозировании объемов продаж в ритейле

Этапы предобработки и описательного анализа данных :

3. Удаление нерелевантных/избыточных признаков:

```
df=df.drop(['Дата', 'День_недели'],axis=1)
```

4. Нормализация целевой переменной - «нормализация по миллиону» (деление на 1 000 000):

```
df['РТО,_руб._без_НДС']=  
df['РТО,_руб._без_НДС']/1000000
```

5. Обработка выбросов:

```
sb.displot(df,x='РТО,_руб._без_НДС',kde=True,  
bins=50)  
sb.boxplot(df,x='РТО,_руб._без_НДС')  
df=df[df['РТО,_руб._без_НДС']<5]
```

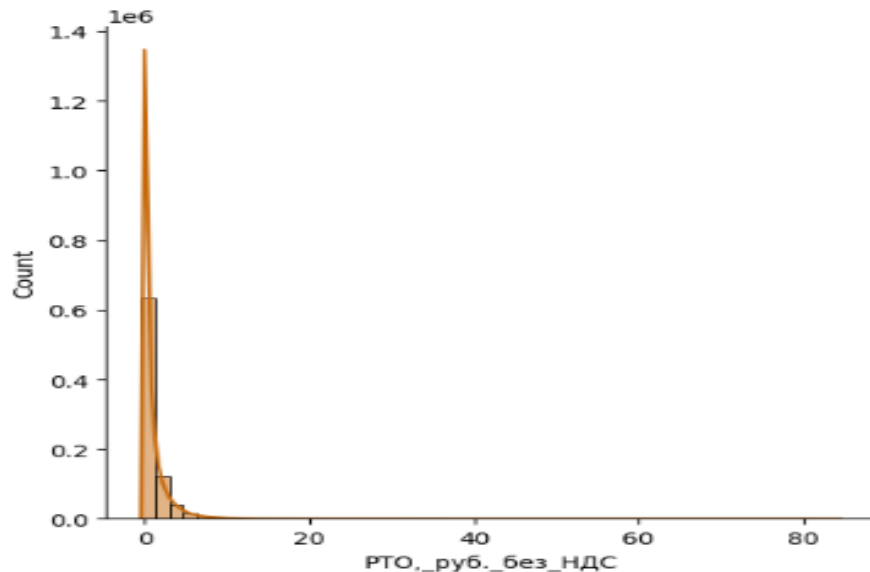


Рис.7 – Гистограмма распределения целевой переменной

Практическое использование алгоритмов ML в прогнозировании объемов продаж в ритейле

Этапы предобработки и описательного анализа данных :

6. Визуализация данных.

7. Сэмплирование (выборочный отбор) используется метод Stratified Sampling:

```
stratified_sample = df.groupby(['Код_Региона',  
'Признак_Промо', 'Признак_ЧТМ', 'Группа_Закупок',  
'Месяц']).apply(  
    lambda x:  
    x.sample(frac=0.01)).droplevel([0,1,2,3,4])
```

```
stratified_sample.shape  
  
(12463, 9)
```

Рис.8 – Размерность стратифицированной выборки

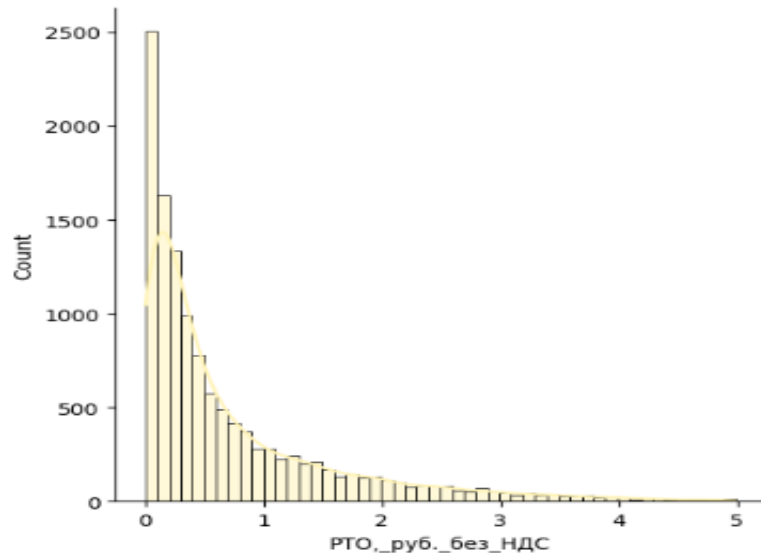


Рис.8 – Гистограмма распределения целевой переменной стратифицированной выборки

Практическое использование алгоритмов ML в прогнозировании объемов продаж в ритейле

Этапы предобработки и описательного анализа данных :

8. Кодирование категориальных переменных:

```
stratified_sample=pd.get_dummies(stratified_sample)
```

9. Разделение данных на обучающую и тестовую выборки:

```
X=stratified_sample.drop(['PTO, _руб._без_НДС'],axis=1)  
Y=stratified_sample['PTO, _руб._без_НДС']  
x_train,x_test,y_train,y_test=train_test_split(X,Y,tes  
t_size=0.05,random_state=22)
```

10. Нормализация данных:

```
scaler= StandardScaler()  
x_train=scaler.fit_transform(x_train)  
x_test=scaler.transform(x_test)
```

Практическое использование алгоритмов ML в прогнозировании объемов продаж в ритейле

Возможно ли использование линейных моделей при экспоненциальном, а не нормальном распределении целевой переменной?

Учебники и книги ниже описывают предпосылки использования линейных моделей:

1. "**An Introduction to Statistical Learning**" by Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani.
Глава 3: Linear Regression. В этой главе обсуждаются основы модели линейной регрессии и рассматриваются предпосылки.
2. "**Applied Linear Statistical Models**" by Michael H. Kutner, Christopher J. Nachtsheim, John Neter, and William Li.
Глава 2: Simple Linear Regression. В этой главе представлены основы простой линейной регрессии, включая предпосылки модели.
3. "**Linear Regression Analysis**" by George A. F. Seber and Alan J. Lee.
Глава 2: The Simple Linear Regression Model. В этой главе рассматриваются предпосылки простой линейной регрессии.

Практическое использование алгоритмов ML в прогнозировании объемов продаж в ритейле

Возможно ли использование линейных моделей при экспоненциальном, а не нормальном распределении целевой переменной?

Нормальность целевой переменной не является необходимым условием для использования модели линейной регрессии и в реальных данных целевые переменные очень часто имеют разнообразные распределения, что не мешает линейно регрессии быть использованной для аппроксимации этих данных.

НО !

Чтобы сделать вывод о пригодности использования линейной модели для конкретных данных, ошибки/остатки проверяются на нормальность распределения и гомоскедастичность!

Проверим ошибки линейной модели на эти два параметра с помощью функции OLS (Ordinary Least Squares – метод наименьших квадратов) :

```
import statsmodels.api as sm
x = sm.add_constant
(stratified_sample.drop
(['PTO, _руб._без_НДС'], axis=1))
Y=stratified_sample['PTO, _руб._без_НДС']
model = sm.OLS(Y, x)
results = model.fit()
residuals = results.resid
```

Практическое использование алгоритмов ML в прогнозировании объемов продаж в ритейле

Проверка ошибок модели линейной регрессии

```
plt.hist(residuals, bins='auto')
plt.xlabel('Остатки')
plt.ylabel('Частота')
plt.title('Гистограмма распределения остатков')
plt.show()
```

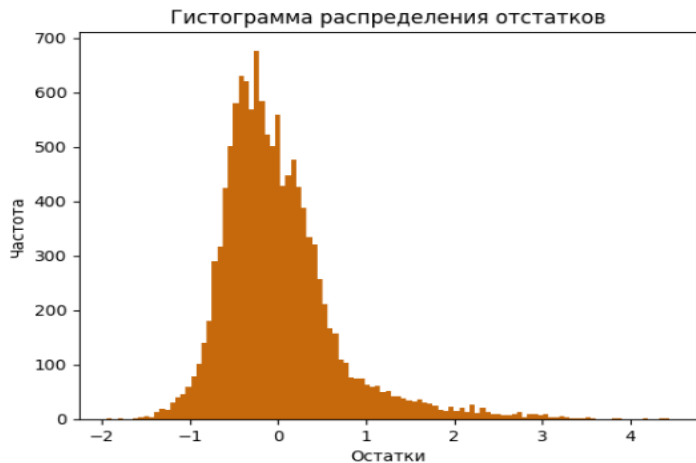


Рис.9 – Гистограмма распределения ошибок

```
# Shapiro-Wilk Test
import scipy.stats as stats
_, p_value = stats.shapiro(residuals)
alpha = 0.05 # Significance level
print(p_value)
if p_value > alpha:
    print("Остатки имеют нормальное распределение")
else:
    print("Распределение остатков не является нормальным")
```

0.0
Распределение остатков не является нормальным

```
#Jarque-Bera Test
from scipy import stats
result = stats.jarque_bera(residuals)
test_statistics = result[0]
p_value=result[1]
alpha = 0.05
print(p_value)
if p_value > alpha:
    print("Остатки имеют нормальное распределение")
else:
    print("Распределение остатков не является нормальным")
```

0.0
Распределение остатков не является нормальным

Практическое использование алгоритмов ML в прогнозировании объемов продаж в ритейле

Проверка ошибок модели линейной регрессии

```
from statsmodels.stats.diagnostic import het_breuschpagan, het_white
# Тест Бройша-Пагана
bp_test = het_breuschpagan(residuals, X)
bp_p_value = bp_test[1]
# Тест Уайта
white_test = het_white(residuals, X)
white_p_value = white_test[1]
print("Breusch-Pagan Test:")
print("p-value:", bp_p_value)

print("White Test:")
print("p-value:", white_p_value)
```

```
Breusch-Pagan Test:
p-value: 5.2398196666051619e-299
White Test:
p-value: 0.0
```

Вывод:

Гипотезы о нормальности распределения ошибок модели линейной регрессии и их гомоскедастичности отвергаются!

Что может помочь стабилизировать дисперсию остатков и приблизить их к нормальному распределению?

Преобразование целевой переменной!

Практическое использование алгоритмов ML в прогнозировании объемов продаж в ритейле

Преобразования целевой переменной

```
sb.displot(stratified_sample,x=np.log(stratified_sample['PTQ',_pyB._6eя_HдC'].to_list()),kde=True, bins=50)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fea63fd110>
```

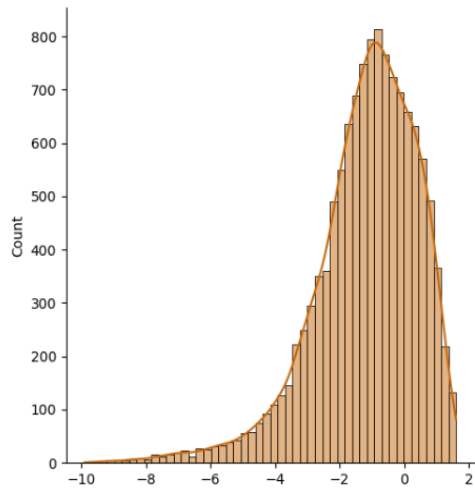


Рис.10 – Гистограмма распределения целевой переменной после логарифмического преобразования

```
sb.displot(stratified_sample,x=np.sqrt(stratified_sample['PTQ',_pyB._6eя_HдC'].to_list()),kde=True, bins=50)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fea5bbc0850>
```

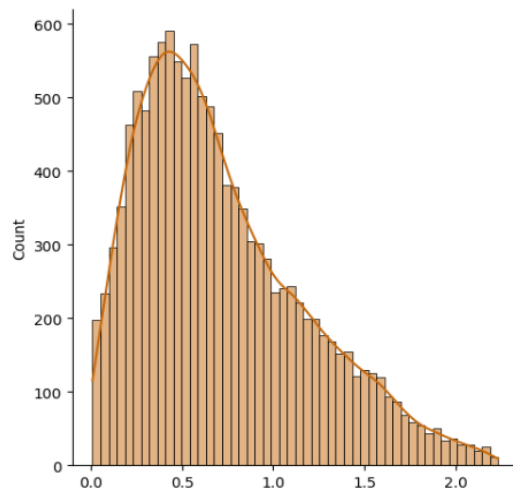


Рис.11 – Гистограмма распределения целевой переменной после преобразования посредством извлечения квадратного корня

Практическое использование алгоритмов ML в прогнозировании объемов продаж в ритейле

Оценка ошибок на нормальность после преобразования целевой переменной

```
import statsmodels.api as sm
x = sm.add_constant(stratified_sample.drop(['РТО', 'руб.', 'без НДС'],axis=1))
Y=np.log(stratified_sample['РТО', 'руб.', 'без НДС'].to_list())
model = sm.OLS(Y, x)
results = model.fit()
residuals = results.resid
```

```
p_value = stats.shapiro(residuals)
p_value
```

```
/usr/local/lib/python3.10/dist-packages/scipy/stats/_morestats.py:1816: UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
ShapiroResult(statistic=0.9395847320556641, pvalue=0.0)
```

```
import statsmodels.api as sm
x = sm.add_constant(stratified_sample.drop(['РТО', 'руб.', 'без НДС'],axis=1))
Y=np.sqrt(stratified_sample['РТО', 'руб.', 'без НДС'].to_list())
model = sm.OLS(Y, x)
results = model.fit()
residuals = results.resid
```

```
p_value = stats.shapiro(residuals)
p_value
```

```
/usr/local/lib/python3.10/dist-packages/scipy/stats/_morestats.py:1816: UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
ShapiroResult(statistic=0.9683968424797058, pvalue=1.401298464324817e-45)
```

Выводы:

Преобразование целевой переменной не помогло улучшить результаты статистических проверок ошибок на нормальность и гомоскедастичности.

Откажемся от использования линейных моделей.

Практическое использование алгоритмов ML в прогнозировании объемов продаж в ритейле

Тестировании моделей нелинейной регрессии.

```
models = [RandomForestRegressor(),
          KNeighborsRegressor(),
          DecisionTreeRegressor(),
          GradientBoostingRegressor(),
          XGBRegressor()]
for i in range(len(models)):
    start=time.time()
    models[i].fit(x_train,y_train)
    print(f'{models[i]} : ')
    y_pred=models[i].predict(x_test)
    end=time.time()
    print(f'r2: {round(r2_score(y_test,y_pred),2)}')
    print(f'rmse: {round(mse(y_test,y_pred)**0.5),2}')
    print(f'time_spent: {round(end-start,2)}')
    print()
```

Выбор топ-3 моделей для прогноза.

На основании полученных результатов составлен short-list рабочих моделей:

I. XGBRegressor() :

r2: 0.78

rmse: 0.42

time_spent: 3.31

II. RandomForestRegressor() :

r2: 0.77

rmse: 0.43

time_spent: 9.89

III. KNeighborsRegressor() :

r2: 0.74

rmse: 0.46

time_spent: 0.15

Практическое использование алгоритмов ML в прогнозировании объемов продаж в ритейле

Подбор гиперпараметров

```
XGBR = XGBRegressor()
params = { 'n_estimators': [100, 200, 300],
          'max_depth': [9, 10, 11],
          'tree_method': ['auto', 'hist'],
          'objective' : ['reg:squarederror', 'reg:squaredlogerror']
        }
grid = GridSearchCV(XGBR, params, cv=2, n_jobs=5, verbose=True)
grid.fit(x_train, y_train)
grid.best_params_
```

```
Fitting 2 folds for each of 36 candidates, totalling 72 fits
{'max_depth': 9,
 'n_estimators': 300,
 'objective': 'reg:squaredlogerror',
 'tree_method': 'hist'}
```

```
XGBR = XGBRegressor(n_estimators=300, max_depth=9, tree_method='hist', objective='reg:squaredlogerror')
XGBR.fit(x_train, y_train, verbose=False)
```

```
start=time.time()
y_pred=XGBR.predict(x_test)
end=time.time()
print(f'r2: {r2_score(y_test, y_pred)}')
print(f'rmse: {mse(y_test, y_pred)**0.5}')
print(f'time_spent: {end-start}')
```

```
r2: 0.7963345112317437
rmse: 0.4135016229156928
time_spent: 0.012389898300170898
```

Выбор лучшей модели

Стратифицированная выборка составила лишь 1,5 % от всего объема данных, 5 % из которых – тестовый набор.

Теперь проверим работоспособность моделей на всем скоупе данных с учетом подобранных GridSearch параметров и выберем лучшую:

XGBRegressor

r2: 0.79

rmse: 0.45



Выводы

Для конкретного набора данных нелинейные модели в целом показали высокую предсказательную способность ($R^2 \approx 0.8$)

Все эти модели предлагает множество параметров и настроек, которые можно оптимизировать для достижения лучшей производительностью.

В результате подбора гиперпараметров самую высокую точность прогнозирования показала бустинговая модель, за счет своей итерационной природы, универсальности и адаптивности к различным типам данных.

A close-up photograph of a hand holding a bright red bell pepper. The hand is positioned inside a metal shopping cart, which is filled with various fresh produce, including green leafy vegetables and a netted bag of green onions. The background is blurred, showing shelves of a grocery store. The overall scene is brightly lit, emphasizing the freshness of the food.

**СПАСИБО
ЗА ВНИМАНИЕ!**



Высшая инженерная школа СПбПУ