

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных систем

Факультет автоматики и вычислительной техники

Кафедра систем автоматизации управления

Предсказание продаж

Отчет по лабораторной работе №3

по дисциплине

«Теория информации, данные, знания»

Выполнил студент гр. ИТб-4302-02-00 _____ /Вершинин П.А./
(Подпись)

Руководитель к.т.н., доцент _____ /Чистяков Г.А./
(Подпись)

Работа защищена с оценкой «_____» «__» _____ 2023 г.

Киров 2023

1 Описание лабораторной работы

Цель: получение студентами навыков работы с прогнозированием временных рядов используя машинное обучение для прогнозирования продаж продуктов питания в магазинах.

В ходе работы необходимо выполнить следующее задание:

- импорт библиотек и загрузка данных;
- подготовка данных;
- обучение и предсказание;
- создание файла для оценки результата.

2 Задание

Спрогнозировать продажи тысяч семейств товаров, продаваемых в магазинах «Favorita», расположенных в Эквадоре. Данные обучения включают даты, информацию о магазине и продукте, рекламировался ли этот товар, а также цифры продаж.

Так же доступны дополнительная информация в следующих файлах:

- «stores.csv» – метаданные магазина, включающие в себя город, штат, тип и кластер;
- «oil.csv» – дневная цена на нефть, включающая значения как за период подготовки, так и за период тестирования данных;
- «holidays_events.csv» – праздники и события с метаданными.

Предсказание необходимо отобразить в файле, который содержит заголовок и иметь следующий формат, представленный на рисунке 1.

```
id, sales
3000888, 0.0
3000889, 0.0
3000890, 0.0
3000891, 0.0
3000892, 0.0
etc.
```

Рисунок 1 – Формат данных в конечном виде

Для каждого идентификатора в тестовом наборе необходимо предсказать значение переменной «sales».

3 Импорт библиотек и загрузка данных

Для начала работы необходимо импортировать следующие библиотеки: «Pandas», «scikit-learn» (sklearn), «Numpy», «XGBRegressor», «statsmodels», «CatBoostRegressor» выполнив следующий код:

```
import numpy as np
import pandas as pd
import datetime
import re
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_log_error
from sklearn.ensemble import GradientBoostingRegressor
from catboost import CatBoostRegressor
from xgboost import XGBRegressor
from statsmodels.graphics.tsaplots import plot_pacf
```

«Numpy» предоставляет поддержку для работы с многомерными массивами и матрицами, а также большой набор математических функций для операций с этими структурами данных.

«XGBRegressor» (eXtreme Gradient Boosting) – это библиотека для машинного обучения, основанная на алгоритме градиентного бустинга. Её разработал Tianqi Chen, и она предоставляет высокоэффективные и масштабируемые реализации градиентного бустинга. Использует настраиваемую функцию потерь, которая включает компоненты для минимизации ошибки и штрафы за сложность модели.

Для загрузки данных из файлов «train.csv» и «test.csv» выполним следующий код:

```
train = pd.read_csv('../train.csv', index_col='id', parse_dates=['date'])

test = pd.read_csv('../test.csv', index_col='id', parse_dates=['date'])

oil = pd.read_csv('../oil.csv', parse_dates=['date'],
index_col='date').to_period('D')

stores = pd.read_csv('../stores.csv', index_col='store_nbr')

holiday = pd.read_csv('../holidays_events.csv')
```

4 Подготовка данных

Перед анализом данных необходимо обработать временные данные («train.csv»), данные о нефти («oil.csv»), данные о праздниках («holidays_events.csv»), данные о магазине («stores.csv»). Обработаем тренировочные данные, создав временные признаки из столбца даты, выполнив следующий код:

```
train = pd.read_csv('../train.csv', index_col='id', parse_dates=['date'])

train_test.date = pd.to_datetime(train_test.date)
train_test['year'] = train_test.date.dt.year
train_test['month'] = train_test.date.dt.month
```

```

train_test['dayofmonth'] = train_test.date.dt.day
train_test['dayofweek'] = train_test.date.dt.dayofweek
train_test['dayname'] = train_test.date.dt.strftime('%A')
)

```

Далее, создадим признак, представляющий собой скользящее среднее за 7 дней для цен на нефть «dcoilwtico». Этот признак используется для анализа тенденций в ценах на нефть, выполнив следующий код:

```

oil['avg_oil_7'] = oil['dcoilwtico'].rolling(7).mean()

trends = pd.DataFrame(index=pd.date_range('2013-01-01', '2017-08-31')).to_period('D')

trends = trends.join(oil, how='outer')
trends['avg_oil_7'].fillna(method='ffill', inplace=True)
trends.dropna(inplace=True)

```

В этом участке кода рассматривается частичная автокорреляция (рис. 2) и создаются лаги для признака «avg_oil_7»:

```

_ = plot_pacf(trends.avg_oil_7, lags=12)

n_lags = 3
for lag in range(1, n_lags + 1):
    trends[f'oil_lags7_{lag}'] = trends.avg_oil_7.shift(lag)
trends.dropna(inplace=True)
trends['date_str'] = trends.index.astype(str)
trends.drop('dcoilwtico', axis=1, inplace=True)

```

Лаги представляют собой значения, сдвинутые на определенное количество временных периодов, и используются для учета влияния предыдущих значений на текущие.

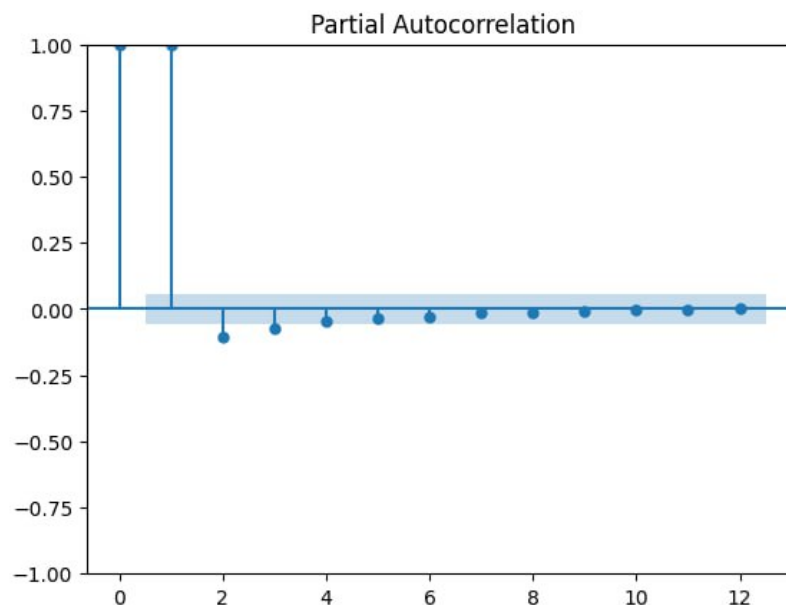


Рисунок 2 – Частичная автокорреляция

Поскольку, в данных отсутствуют цены на некоторых промежутках времени, используем линейную интерполяцию для заполнения пропущенных значений в данных о нефти выполнив код:

```
oil = oil.interpolate(method='linear')
oil.iloc[0] = oil.iloc[1]

start_date = train_test.date.min()
number_of_days = 1704
date_list = [(start_date + datetime.timedelta(days=day)).isoformat() for
day in range(number_of_days)]

date = (pd.Series(date_list)).to_frame()
date.columns = ['date']
date.date = pd.to_datetime(date.date)
date['date_str'] = date.date.astype(str)
oil['date_str'] = oil.index.astype(str)

oil = pd.merge(date, oil, how='left', on='date_str')
oil =
```

```
oil.set_index('date').dcoilwtico.interpolate(method='linear').to_frame()  
oil['date_str'] = oil.index.astype(str)
```

Создадим два новых признака «onpromo_7» и «onpromo_28», связанных с акциями, выполнив код ниже. Эти признаки представляют собой скользящее среднее за 7 и 28 дней для бинарного признака «onpromotion». Если данный признак равен 1 (акция проводится), то соответствующее значение в новом признаке будет средним значением за последние 7 и 28 дней, в зависимости от признака.

```
train_test['onpromo_7'] = train_test['onpromotion'].rolling(7).mean()  
train_test['onpromo_28'] = train_test['onpromotion'].rolling(28).mean()  
train_test['onpromo_7'].fillna(0, inplace=True)  
train_test['onpromo_28'].fillna(0, inplace=True)
```

Далее, создадим несколько признаков, связанных с праздниками:

- «national_holiday» – 1, если это национальный выходной, 0 в противном случае;
- «national_event» – 1, если это национальное мероприятие, 0 в противном случае;
- «national_workday» – 1, если это национальный рабочий день, 0 в противном случае;
- «local_holiday» – локальный праздник, который происходит в городе;
- «regional_holiday» – региональный праздник;
- «weekend» – 1, если это выходной день (день недели больше или равен 5), 0 в противном случае.

Применяем логарифм к данным о продажах «train_test.sales», чтобы уменьшить дисперсию и асимметрию, выполнив следующий код:

```

holiday = holiday.loc[holiday['transferred'] == False]
holiday['description'] = holiday['description'].str.replace('Traslado ',
'')

day_off = holiday.where((holiday['type'] != 'Work Day') | (holiday['type']
!= 'Event')).set_index('date')['description'].to_dict()
train_test['date_str'] = train_test.date.astype(str)
train_test['national_holiday'] = [1 if a in day_off else 0 for a in
train_test.date_str]

event = holiday.where(holiday['type'] ==
'Event').set_index('date')['description'].to_dict()
train_test['national_event'] = [1 if a in event else 0 for a in
train_test.date_str]

work_day = holiday.where(holiday['type'] == 'Work
Day').set_index('date')['description'].to_dict()
train_test['national_workday'] = [1 if a in work_day else 0 for a in
train_test.date_str]

train_test['weekend'] = [1 if a >= 5 else 0 for a in train_test.dayofweek]

# Local and Regional
local = holiday.loc[holiday['locale'] == "Local"]
local_dic = local.set_index('date')['locale_name'].to_dict()
train_test['local_holiday'] = [1 if b in local_dic and local_dic[b] == a
else 0 for a, b in zip(train_test.city, train_test.date_str)]

regional = holiday.loc[holiday['locale'] == "Regional"]
regional_dic = regional.set_index('date')['locale_name'].to_dict()
train_test['regional_holiday'] = [1 if b in regional_dic and
regional_dic[b] == a else 0 for a, b in zip(train_test.state,
train_test.date_str)]

train_test.sales = np.log1p(train_test.sales)

```


Лаги представляют собой отстающие значения временного ряда, которые могут быть использованы в качестве признаков для прогнозирования будущих значений, выполнив следующий код:

```
train_test['Lag_7'] = train_test['sales'].shift(1782 * 7)

for i in range(16, 24):
    train_test['Lag_{i}'] = train_test['sales'].shift(1782 * i)

train_test['Lag_28'] = train_test['sales'].shift(1782 * 28)
train_test['Lag_30'] = train_test['sales'].shift(1782 * 30)
train_test['Lag_31'] = train_test['sales'].shift(1782 * 31)
train_test['Lag_365'] = train_test['sales'].shift(1782 * 365)
```

Следующая обработка, включает создание экспоненциально взвешенных скользящих средних «EWMA» для различных альфа и лагов, категориальных признаков, выполнив следующий код:

```
def ewm_features(dataframe, alphas, lags):
    for alpha in alphas:
        for lag in lags:
            feature_name = f'ewm_{alpha}_{lag}'
            dataframe[feature_name] = dataframe.groupby(['store_nbr',
                                                         'family'])['sales'].transform(
                lambda x: x.shift(lag).ewm(alpha=alpha,
                                              min_periods=1).mean())
    return dataframe

alphas = [0.95, 0.8, 0.65, 0.5]
lags = [1, 7, 30]
train_test = ewm_features(train_test, alphas, lags)

categories = ['city', 'state', 'type']
for category in categories:
    encoder = preprocessing.LabelEncoder()
    train_test[category] = encoder.fit_transform(train_test[category])
```

«LabelEncoder» из библиотеки «scikit-learn» для преобразования категориальных признаков (город, штат, тип) в числовые значения.

5 Обучение и предсказание

В процессе разработки прогностической модели для задачи прогнозирования временных рядов, было принято решение использовать «XGBRegressor» и «CatBoostRegressor» в качестве основных моделей. Этот выбор обосновывается рядом ключевых факторов, целью которых было обеспечить высокую точность прогнозов, эффективность обучения и удобство работы с особенностями данных.

Объединение обеих моделей позволяет учесть различия в поведении продаж для различных семейств продуктов, что важно для создания точной и устойчивой модели прогнозирования. Такой подход позволяет совместить преимущества обеих моделей и достичь более высокой общей производительности, что отражено в качестве прогнозов на конечном этапе разработки модели.

Цикл обучения и прогнозирования для каждого семейства продуктов «family» в наборе данных, используем следующий код:

```
cat = CatBoostRegressor()
xgb = XGBRegressor(**params)
for family in families:
    train_family = train.loc[train['family'] == family]
    X_train_family, X_val_family, y_train_family, y_val_family =
train_test_split(train_family, train_family[TARGET],
test_size=0.05, shuffle=False)

    cat.fit(
        X_train_family[FEATURES], y_train_family,
eval_set=[(X_train_family[FEATURES], y_train_family), (X_val_family[FEATURES],
y_val_family)], verbose=False, early_stopping_rounds=10
    )

    xgb.fit(
```

```
X_train_family[FEATURES], y_train_family,  
eval_set=[(X_train_family[FEATURES], y_train_family),  
(X_val_family[FEATURES], y_val_family)], verbose=False  
)
```

Данные разделяются в пропорции 95/5 для обучения и валидации соответственно. Далее, обе модели обучаются на данных, используя обучающую выборку и оцениваются на валидационной выборке. «early_stopping_rounds» прекращает обучение «CatBoost», если не происходит улучшение качества модели после определенного числа итераций.

Предсказания и фактические значения расширяются для последующей оценки общего качества модели на всем наборе данных. Каждая модель используется для прогнозирования продаж на тестовой выборке для конкретного семейства.

Предсказания для тестовой выборки расширяются для последующего агрегирования предсказаний по всем семействам продуктов. Создается дополнительный фрейм данных, содержащий информацию о продажах для всех продуктов в тестовой выборке.

Создаем новый список предсказаний «predictions», который является линейной комбинацией предсказаний двух моделей. Это достигается путем взвешенного усреднения предсказаний каждой модели. Веса для каждой модели равны 0.5, что означает, что каждая модель вносит одинаковый вклад в окончательные предсказания.

6 Создание файла для оценки результата

Загружаем CSV-файл, который представляет собой образец для представления результатов, выполнив следующий код:

```
output = pd.read_csv('../sample_submission.csv', index_col='id')
output['sales'] = np.expml(test_predict['pred'])
output.to_csv('submission_Store_sales.csv')
```

Указание «index_col='id'» говорит «pandas» использовать столбец «id» в качестве индекса для «DataFrame». Присваиваем столбцу «sales» в фрейм данных «output» значения прогнозов, полученных из переменной «test_predict['pred']». Перед присвоением происходит обратное преобразование значений из логарифмической шкалы в нормальную с использованием «np.expml». Сохраняем полученный результат в CSV-файл и отправляем на оценку.



	submission_without_TSFRESH.csv Complete · 2h ago · XGBoosting && CatBoostBoosting	0.4147
	submissionTFS.csv Complete · 16h ago · xgboosting	2.25429

Рисунок 3 – Результат оценок

Оценочным показателем для данной задачи является среднеквадратичная логарифмическая ошибка. Чем меньше оценка, тем лучше результат в итоге.

Приведенный ниже результат, отображает решение без использования «CatBoostRegressor», равный 2,25429.

Таким образом, одно из решений данной задачи является использование таких библиотек, как «CatBoostRegressor» и «XGBRegressor» с результатом 0,4147.

Вывод

В рамках данной лабораторной работы был проведен анализ временных рядов данных о продажах, с использованием различных методов обработки и преобразования данных. Были рассмотрены факторы, влияющие на продажи, такие

как праздники, тенденции в ценах на нефть, а также использованы методы создания лагов и экспоненциального взвешенного скользящего среднего.