Лабораторная работа №

«Прогнозирование временного ряда в Python с помощью метода Хольта-Уинтерса»

**Цель работы:** получение навыков прогнозирования с помощью метода Хольта-Уинтерса и оценки результатов в Python.

## ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Как мы знаем, временной ряд можно разложить на 3 составляющие: тренд, сезонность и случайная составляющая. Модель Хольта-Уинтерса использует идеи экспоненциального сглаживания, но является более сложной и может применяться к рядам, содержащим тенденцию и сезонность. Она работает по следующим формулам:

$$L_t = \alpha * \frac{Y_t}{S_{t-s}} + (1-\alpha)(L_{t-1} + T_{t-1})$$

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1}$$

$$S_t = \gamma \frac{Y_t}{L_t} + (1-\gamma)S_{t-s}$$

$$\hat{Y}_{t+p} = (L_t + pT_t)S_{t-s+p}$$

где

Lt – новая сглаженная величина;

α – постоянная сглаживания для данных ($0 \leq \alpha \geq 1$);

Yt – новое наблюдение или реальное значение ряда в период t;

β - постоянная сглаживания для оценки тренда ($0 \leq \beta \leq 1$);

Tt - собственно оценка тренда;

(гамма) - постоянная сглаживания для оценки сезонности;

St – оценка сезонности;

s – длительность периода сезонного колебания;

p – количество периодов вперед, на которое делается прогноз;

Ŷ t+p - прогноз на p периодов вперед.

К счастью, модель реализована в одной из библиотек Python и не требуется

программировать все формулы самостоятельно. Прогнозирование можно осуществить используя метод **statsmodels.tsa.holtwinters.ExponentialSmoothing()**.

В данном методе, в зависимости от наличия во временном ряде тренда и сезонности, добавляются или убираются параметры **trend** и **seasonal**, также нужно определить периодичность сезонности параметром **seasonal_periods**.

**Пример 1**

Для начала нужно скачать необходимые компоненты, нам понадобятся следующие библиотеки: **pandas** для упрощения работы с табличными данными, **numpy** для математических операций, **statsmodels** для разбиения временного ряда на компоненты и последующего прогнозирования, а также **sklearn** для оценки построенной модели.

Перед началом работы нам потребуется установить данные библиотеки, это можно сделать с помощью следующих команд:

*!pip install pandas*

*!pip install numpy*

*!pip install statsmodels*

*!pip install sklearn*

В данной работе попробуем спрогнозировать среднюю месячную зарплату в Беларуси. Для этого понадобится скачать данные по ссылке или с официального сайта Белстат. После скачивания необходимо будем провести действия по приведению данных в одинаковый вид (убрать комментарии, привести значения средней заработной платы в вид 2016 г. с учетом всех деноминаций). После данной операции, прочитаем данные с помощью метода **pandas.read_csv()**, поставив первую колонку как индекс и сбросим 2022 год для упрощения последующих вычислений.

*df = pandas.read_csv('salary.csv', index_col=0).drop(2022)*

В итоге наша таблица примет следующий вид:

| Год | Январь | Февраль | Март | Апрель |
|---|---|---|---|---|
| **1991** | 0.0000026600 | 0.0000026800 | 0.0000029500 | 0.0000035800 |
| **1992** | 0.0000125600 | 0.0000150700 | 0.0000211700 | 0.0000244500 |
| **1993** | 0.0001293900 | 0.0001625200 | 0.0001889800 | 0.0002360000 |
| **1994** | 0.0018846900 | 0.0020437900 | 0.0023852500 | 0.0039695900 |
| **1995** | 0.0339666000 | 0.0387739000 | 0.0653853000 | 0.0692137000 |
| **1996** | 0.0950903000 | 0.1015329000 | 0.1066554000 | 0.1062414000 |
| **1997** | 0.1526205000 | 0.1639575000 | 0.1732454000 | 0.1898682000 |
| **1998** | 0.3024216000 | 0.3389950000 | 0.3698217000 | 0.3669287000 |

Сейчас данные представлены в табличном виде, однако для вычислений нам потребуется чтобы данные за каждый месяц шли один за другим.

```
monthly_salaries = []

for row in range(len(df)):
    monthly_salaries.extend(list(df.iloc[row]))
```

Получим следующий вид:

```
0          0.0000026600
1          0.0000026800
2          0.0000029500
3          0.0000035800
4          0.0000040900
              ...
367    1,463.2000000000
368    1,442.7000000000
369    1,478.9000000000
370    1,476.3000000000
371    1,675.3000000000
Length: 372, dtype: object
```

Для того чтобы избавиться от вероятности применения неправильной модели - логарифмируем наши данные (данная операция позволит без опасений применять аддитивную модель разложения временного ряда) - это можно сделать с помощью метода **log()** библиотеки **numpy**. Далее произведем предобработку данных:

- Заменим индекс на Datetime
- Определим параметр frequency как месячный

```
monthly_salaries = pandas.Series(monthly_salaries,
index=pandas.date_range(start='1/1/1991', periods=len(monthly_salaries),
freq='M'))
```

Всего у нас есть 372 наблюдений, для тренировки оставим 350, а для тестирования оставшиеся 22.

*train = monthly_salaries [:350]*
*test = monthly_salaries [350:]*

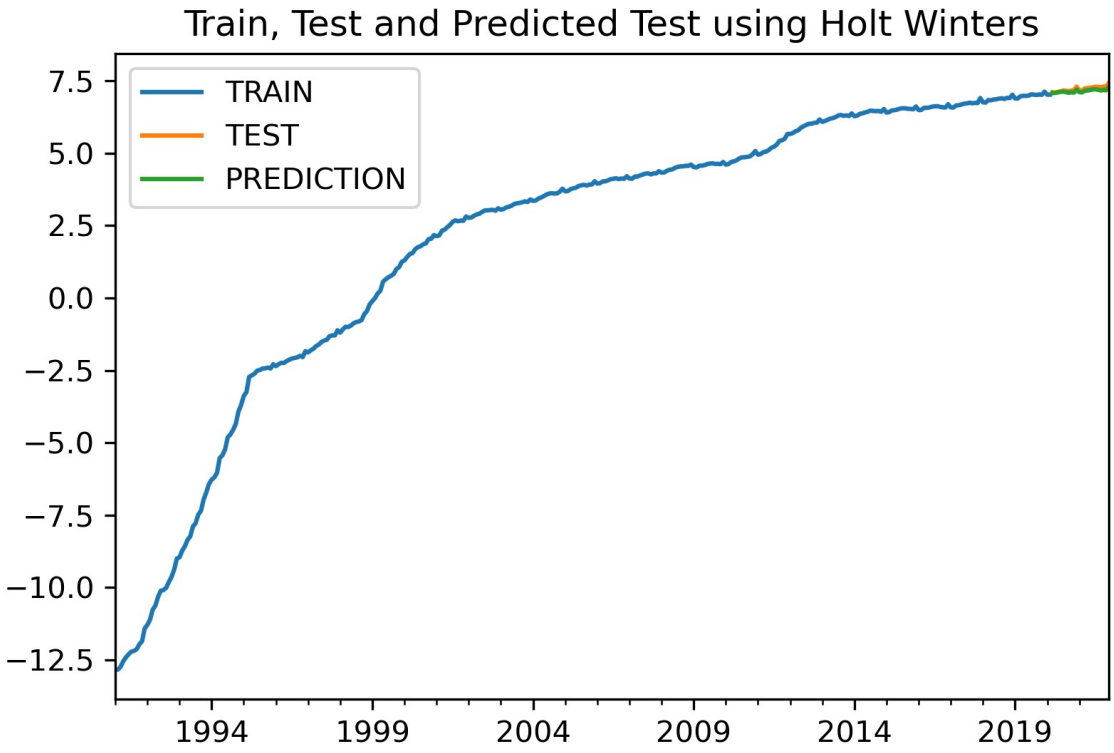
Теперь тренируем нашу модель

*fitted_model = statsmodels.tsa.holtwinters.ExponentialSmoothing(train,trend='add',seasonal='add', seasonal_periods=12).fit()*

Делаем прогноз

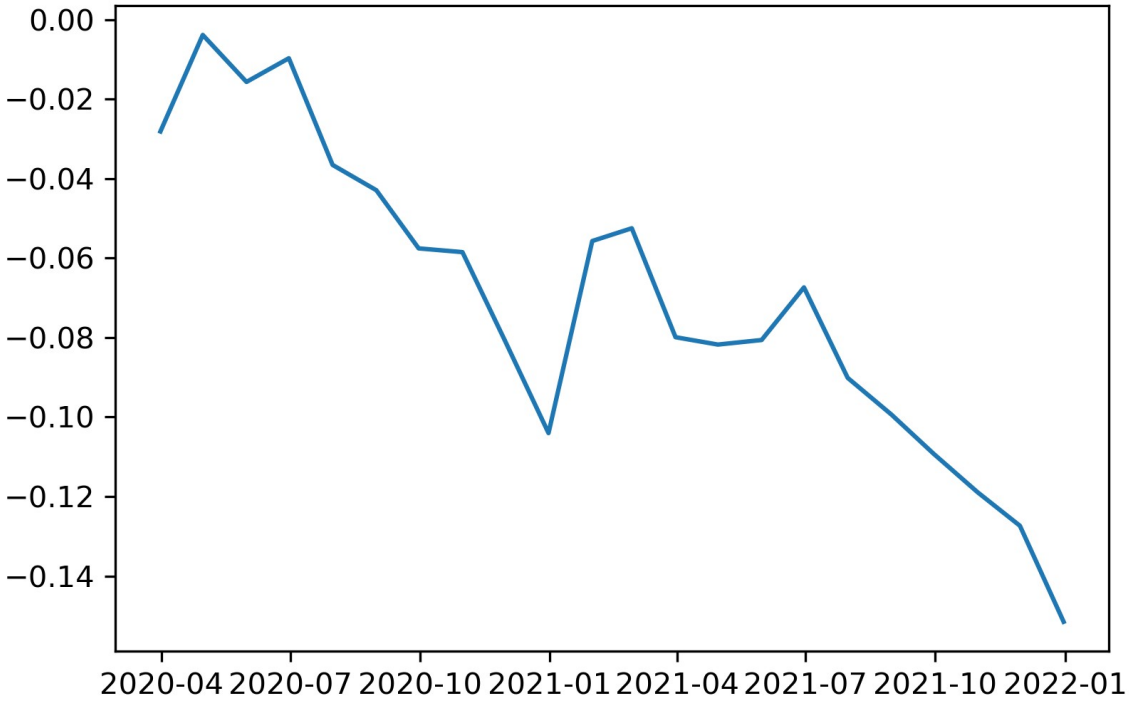
*test_predictions = fitted_model.forecast(len(test))*

Визуализируем результаты




В целом наше предсказание соответствует действительности, теперь займемся его оценкой:

В начале глянем на остатки, ими является разница между нашим предсказанием и реальными данными




Лучшим результатом являются стационарные остатки около нуля, в нашем случае они растут со временем и не являются стационарными, что является очень плохим признаком.

**MSE (Mean Squared Error)**

Следующая мера оценки - средняя квадратичная ошибка, она, в отличие от средней ошибки, наказывает за сильное отклонение от истинного значения. Она считается по следующей формуле:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

$\mathrm{MSE}$ = mean squared error
$n$ = number of data points
$Y_i$ = observed values
$\hat{Y}_i$ = predicted values

Но, так как в нашем случае мы имеем ошибки от 0 до 1 - будет правильно взять **RMSE (Root Mean Squared Error)** для интерпретации результатов. Данный показатель вычисляется, беря квадрат из **MSE**.

Для вычислений используется метод **mean_squared_error()** с параметром **squared=False** (для получения **RMSE**), пакета **sklearn.metrics**.

```
sklearn.metrics.mean_squared_error(test, test_predictions, squared=False)
```

```
0.08022773325133858
```

В нашем случае он равняется **0.08** или около 1% от последних значений, что являлось бы приемлемым значением, если бы наш ряд был стационарен и следующий показатель тоже имел приемлемое значение.

**R-квадрат (Коэффициент детерминации)**

Она показывает, насколько условная дисперсия модели отличается от дисперсии реальных значений Y. Если этот коэффициент близок к 1, то условная дисперсия модели достаточно мала и весьма вероятно, что модель неплохо описывает данные. Если же коэффициент **R-квадрат** сильно меньше, например, меньше 0.5, то, с большой долей уверенности модель не отражает реальное положение вещей. Вычисление можно произвести с помощью метода **r2_score()**, пакета **sklearn.metrics**.

```
sklearn.metrics.r2_score(test, test_predictions)
```

```
0.09488777860417452
```

В нашем случае коэффициент детерминации **~0.095**, что говорит о полном отличии дисперсии предсказанных значений от дисперсии реальных значений.

**Самостоятельно задание:**

Скачайте в интернете набор данных временного ряда и сделайте прогноз на ближайшее время с помощью метода Хольта-Винтерса, производите оценку своего прогноза и интерпретируйте её.

**Контрольные вопросы:**

1. Какие еще есть методы прогнозирования временных рядов?
2. В каком случае высокий коэффициент детерминации не говорит о низком отличии реальной и предсказанной дисперсии?
3. Перечислите способы оценки методов классификации в машинном обучении и объясните как их интерпретировать.