

▼ Лабораторная работа №6

Цель лабораторной работы: изучение основных методов анализа и прогнозирование временных рядов.

Задание:

1. Выберите набор данных (датасет) для решения задачи прогнозирования временного ряда.
2. Визуализируйте временной ряд и его основные характеристики.
3. Разделите временной ряд на обучающую и тестовую выборку.
4. Произведите прогнозирование временного ряда с использованием как минимум двух методов.
5. Визуализируйте тестовую выборку и каждый из прогнозов.
6. Оцените качество прогноза в каждом случае с помощью метрик.

▼ Импорт библиотек

```
import numpy as np
import pandas as pd
from matplotlib import pyplot
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from statsmodels.tsa.arima_model import ARIMA
from sklearn.model_selection import GridSearchCV
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
```

▼ Загрузка данных

Монтирование Google Drive для получения доступа к данным, лежащим на нем:

```
from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m



Загрузка данных:

```
filename = '/content/drive/MyDrive/POP.csv'
```

```
data = pd.read_csv(filename, sep=',')
```


```
data = data.drop(['realtime_start', 'realtime_end'], axis=1)
```

```
"""Преобразование столбца даты в объект datetime и установка его в качестве индекса"""
```


```
data['date'] = pd.to_datetime(data['date'])
```

```
data.set_index('date', inplace=True)
```

```
data.head()
```

	value 
date	
1952-01-01	156309.0
1952-02-01	156527.0
1952-03-01	156731.0
1952-04-01	156943.0
1952-05-01	157140.0

```
data.describe()
```

	value 
count	816.000000
mean	243847.767826
std	50519.140567
min	156309.000000
25%	201725.250000
50%	239557.500000
75%	289364.250000
max	330309.946000

В качестве датасета будем использовать набор данных, содержащий данные для прогнозирования цен на акции компании Netflix.

<https://www.kaggle.com/datasets/ranugadisansagamage/netflix-stocks>

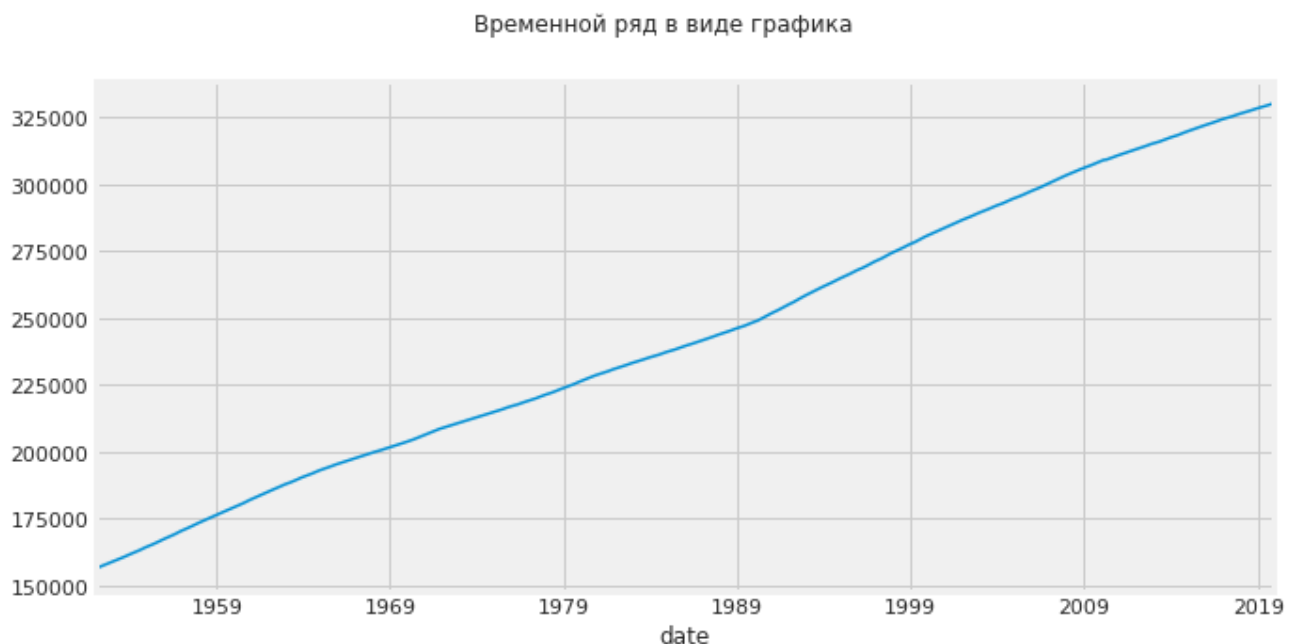
Датасет содержит следующие атрибуты:

1. Date - The day - Дата
2. Open - The open Price of the Stock - цена в день открытия акции

3. High - The highest price of the stock - самая высокая цена акции
4. Low - The lowest price of the stock - самая низкая цена акции
5. Close - The closing price of the stock - цена акции в день закрытия
6. Adj Close - amends a stock's closing price to reflect that stock's value after accounting for any corporate actions - изменяет цену закрытия акции, чтобы отразить стоимость этой акции после учета любых корпоративных действий
7. Value - объем продаж

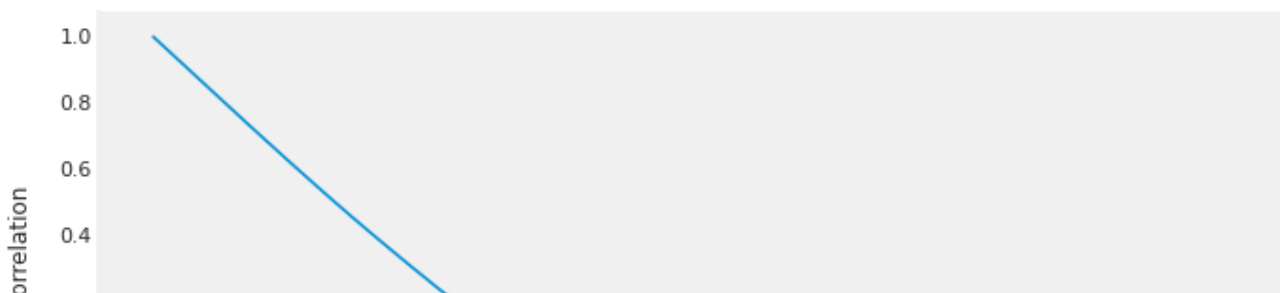
▼ Визуализация временного ряда

```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('Временной ряд в виде графика')
data.plot(ax=ax, legend=False)
pyplot.show()
```



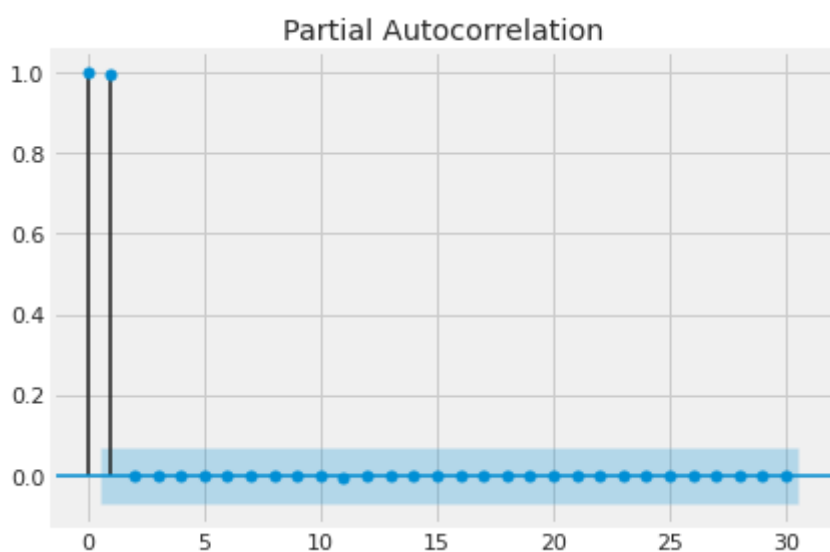
```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('Автокорреляционная диаграмма')
pd.plotting.autocorrelation_plot(data, ax=ax)
pyplot.show()
```

Автокорреляционная диаграмма



▼ Частичная автокорреляционная функция

```
plot_pacf(data, lags=30)
plt.tight_layout()
```



Разделение временного ряда на обучающую и тестовую выборку

```
data2 = data.copy()
```

```
# Целочисленная метка шкалы времени
xnum = list(range(data2.shape[0]))
# Разделение выборки на обучающую и тестовую
Y = data2['value'].values
train_size = int(len(Y) * 0.7)
xnum_train, xnum_test = xnum[0:train_size], xnum[train_size:]
train, test = Y[0:train_size], Y[train_size:]
history_arima = [x for x in train]
history_es = [x for x in train]
```

Прогнозирование временного ряда авторегрессионным методом (ARIMA)

```

from statsmodels.tsa.arma_model import ARIMA

from statsmodels.tsa.holtwinters import ExponentialSmoothing

from sklearn.metrics import mean_squared_error

# Параметры модели (p,d,q)
arma_order = (2,1,0)
# Формирование предсказаний
predictions_arma = list()
for t in range(len(test)):
    model_arma = ARIMA(history_arma, order=arma_order)
    model_arma_fit = model_arma.fit()
    yhat_arma = model_arma_fit.forecast()[0]
    predictions_arma.append(yhat_arma)
    history_arma.append(test[t])
# Вычисление метрики RMSE
error_arma = mean_squared_error(test, predictions_arma, squared=False)

# Ошибка прогноза
np.mean(Y), error_arma

(243847.7678259804, 7372.900036604332)

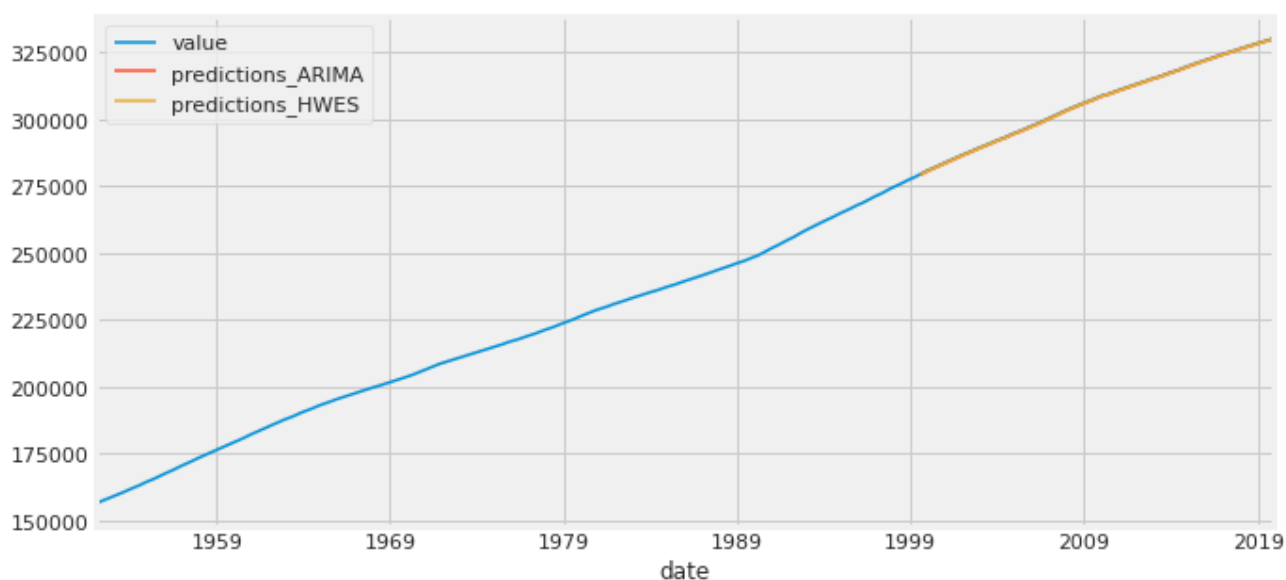
# Формирование предсказаний
predictions_es = list()
for t in range(len(test)):
    model_es = ExponentialSmoothing(history_es)
    model_es_fit = model_es.fit()
    yhat_es = model_es_fit.forecast()[0]
    predictions_es.append(yhat_es)
    history_es.append(test[t])
# Вычисление метрики RMSE
error_es = mean_squared_error(test, predictions_es, squared=False)

# Записываем предсказания в DataFrame
data2['predictions_ARIMA'] = (train_size * [np.NaN]) + list(predictions_arma)
data2['predictions_ARIMA'] = (train_size * [np.NaN]) + list(predictions_es)

fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('Предсказания временного ряда')
data2.plot(ax=ax, legend=True)
pyplot.show()

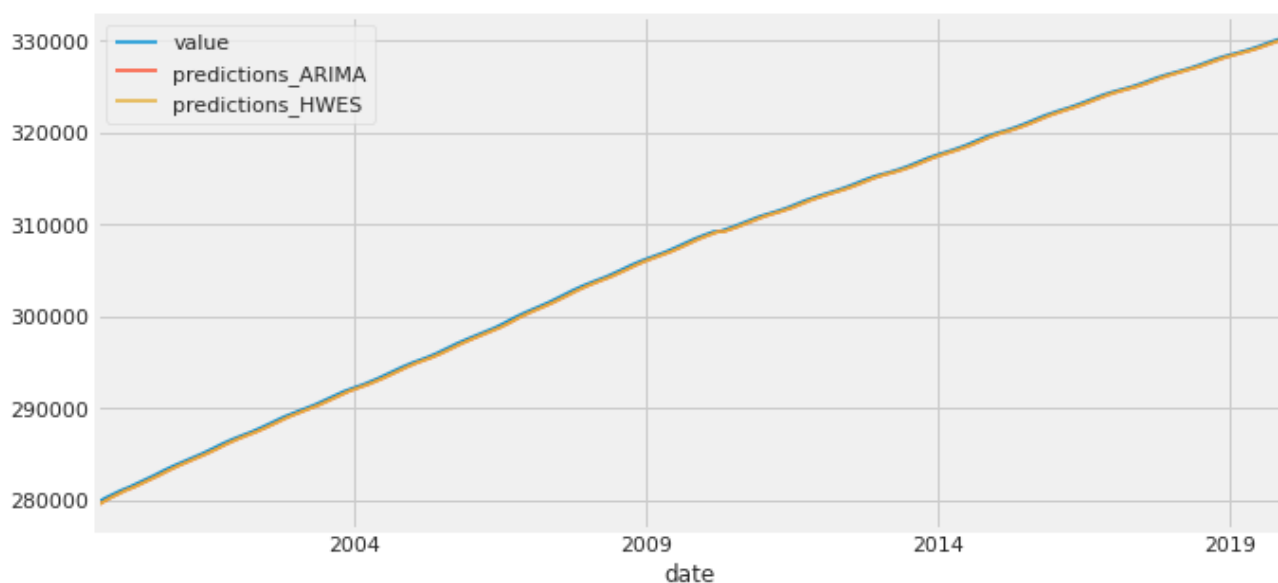
```

Предсказания временного ряда



```
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))
fig.suptitle('Предсказания временного ряда (тестовая выборка)')
data2[train_size:].plot(ax=ax, legend=True)
pyplot.show()
```

Предсказания временного ряда (тестовая выборка)



▼ Прогнозирование временного ряда методом символьной регрессии

```
pip install gplearn
```

Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/>

Collecting gplearn

Downloading gplearn-0.4.2-py3-none-any.whl (25 kB)

Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.7/dist-packages

Requirement already satisfied: joblib>=1.0.0 in /usr/local/lib/python3.7/dist-packages

Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages

Requirement already satisfied: numpy>=1.14.6 in /usr/local/lib/python3.7/dist-packages

Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages

Installing collected packages: gplearn

Successfully installed gplearn-0.4.2

```
from gplearn.genetic import SymbolicRegressor
```

```
function_set = ['add', 'sub', 'mul', 'div', 'sin']
```

```
SR = SymbolicRegressor(population_size=500, metric='mse',
                       generations=70, stopping_criteria=0.01,
                       init_depth=(4, 10), verbose=1, function_set=function_set,
                       const_range=(-100, 100), random_state=0)
```

```
SR.fit(np.array(xnum_train).reshape(-1, 1), train.reshape(-1, 1))
```

Population Average			Best Individual			
Gen	Length	Fitness	Length	Fitness	OOB Fitness	Time Le
0	263.65	2.43463e+63	23	7.14077e+09	N/A	2.9
1	130.98	5.77055e+16	43	6.06688e+09	N/A	1.2
2	53.10	4.58992e+15	34	3.54847e+09	N/A	44.30
3	34.28	1.99853e+19	13	1.42699e+09	N/A	34.84
4	35.05	2.10424e+16	38	1.04052e+09	N/A	35.3
5	30.47	2.56729e+16	36	4.29436e+08	N/A	32.7
6	31.30	3.00498e+16	50	6.39791e+07	N/A	32.0
7	38.37	8.59782e+15	35	1.51165e+07	N/A	33.3
8	43.37	5.29474e+15	47	4.76034e+06	N/A	33.0
9	37.70	8.42452e+15	35	4.14545e+06	N/A	31.2
10	40.68	5.69103e+15	32	3.65059e+06	N/A	32.44
11	45.38	5.71108e+15	29	3.65015e+06	N/A	33.6
12	41.36	5.72894e+15	29	3.65015e+06	N/A	30.9
13	35.07	3.58233e+15	29	3.65015e+06	N/A	28.08
14	33.33	8.46569e+15	35	3.53261e+06	N/A	27.30
15	31.43	3.14997e+19	35	3.53261e+06	N/A	26.1
16	30.19	1.42657e+16	35	3.53261e+06	N/A	24.79
17	30.81	2.81228e+15	35	3.53261e+06	N/A	26.79
18	33.31	5.72757e+15	35	3.53261e+06	N/A	25.70
19	33.71	1.26632e+16	35	3.50395e+06	N/A	24.44
20	34.95	1.70198e+16	35	3.50395e+06	N/A	25.5
21	42.21	6.70957e+15	35	3.50395e+06	N/A	24.69
22	54.68	6.78469e+15	35	3.50395e+06	N/A	26.1
23	50.99	6.47928e+18	102	3.50387e+06	N/A	25.80
24	42.69	8.57551e+15	71	3.50376e+06	N/A	23.80
25	59.07	6.73374e+21	85	3.49756e+06	N/A	24.78
26	89.07	1.51918e+25	85	3.49756e+06	N/A	29.6
27	100.70	2.98833e+18	91	3.48956e+06	N/A	29.39
28	120.58	7.92131e+23	91	3.48956e+06	N/A	30.88
29	142.26	1.91023e+18	127	3.48498e+06	N/A	35.27
30	116.37	6.9315e+21	54	3.46676e+06	N/A	29.78

31	103.96	2.33782e+22	54	3.46676e+06	N/A	32.00
32	107.16	2.82439e+18	54	3.46676e+06	N/A	26.54
33	110.56	4.95099e+26	112	3.45858e+06	N/A	26.21
34	94.20	1.96986e+18	114	3.45249e+06	N/A	24.41
35	77.71	6.0703e+15	133	3.43034e+06	N/A	20.81
36	111.25	5.62717e+15	79	3.42948e+06	N/A	23.21
37	142.44	1.4552e+18	246	3.41658e+06	N/A	26.81
38	171.28	3.11029e+19	187	3.36822e+06	N/A	28.61
39	197.58	2.8446e+16	187	3.36419e+06	N/A	30.50
40	213.08	1.12226e+16	212	3.35931e+06	N/A	29.81
41	193.33	7.07447e+17	181	3.35563e+06	N/A	27.04
42	200.58	9.48793e+19	308	3.25166e+06	N/A	26.31
43	203.16	6.9535e+17	308	3.24914e+06	N/A	25.41
44	271.65	2.48275e+15	434	3.17665e+06	N/A	29.01
45	340.95	1.45248e+18	434	3.17665e+06	N/A	31.61
46	407.23	2.9286e+14	874	3.13466e+06	N/A	36.11
47	475.59	8.20919e+13	857	3.13086e+06	N/A	38.70
48	698.39	6.58531e+17	1124	3.1245e+06	N/A	49.51
49	871.75	5.67064e+14	1140	3.1232e+06	N/A	56.91
50	1008.67	1.44739e+18	1126	3.11533e+06	N/A	1.01
51	1040.20	8.00984e+13	1337	3.1087e+06	N/A	59.61
52	1087.90	4.8939e+10	1352	3.10262e+06	N/A	59.01
53	1212.74	6.88053e+18	1338	3.09244e+06	N/A	1.01

```
# Предсказания
```

```
y_sr = SR.predict(np.array(xnum_test).reshape(-1, 1))
```

```
y_sr[:10]
```

```
array([274891.75793852, 274817.36307349, 275068.42349884, 275594.91553188,
       275909.57465535, 276033.9204471 , 276192.3291504 , 276368.95663777,
       276651.56236873, 276542.01774132])
```

▼ Качество прогноза моделей

```
def print_metrics(y_test, y_pred):
    print(f"R^2: {r2_score(y_test, y_pred)}")
    print(f"MSE: {mean_squared_error(y_test, y_pred, squared=False)}")
    print(f"MAE: {mean_absolute_error(y_test, y_pred)}")
```

```
print("ARIMA")
print_metrics(test, predictions_arima)
```

```
print("\nGPlearn")
print_metrics(test, y_sr)
```

```
ARIMA
R^2: 0.7495397912366841
MSE: 7372.900036604332
MAE: 751.3845663510563
```

```
GPlearn
R^2: 0.8047153645391025
MSE: 6510.330169456957
MAE: 6443.710113418146
```


✓ 0 сек. выполнено в 14:05

