**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Радиотехнический»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №6

«Анализ и прогнозирование временного ряда.»

Выполнил: Проверил:

студент группы РТ5-61Б преподаватель каф. ИУ5

Агеев Алексей Гапанюк Ю.Е.

Подпись и дата: Подпись и дата:

Москва, 2023 г

# Описание задания

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.

# Ход работы

import numpy as np  
import pandas as pd  
from matplotlib import pyplot  
import matplotlib.pyplot as plt

def clear(data):  
 data['humidity'] = None  
 data['meantemp'] = None  
 data['meanpressure'] = None

data\_start = pd.read\_csv('DailyDelhiClimateTrain.csv', header=0, index\_col=0, parse\_dates=True)  
  
clear(data\_start)  
  
data\_test = pd.read\_csv('DailyDelhiClimateTest.csv', header=0, index\_col=0, parse\_dates=True)  
clear(data\_test)  
  
data\_train = data\_start['wind\_speed']

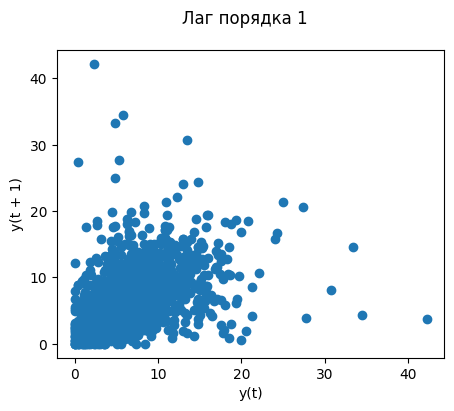
data\_train.head()

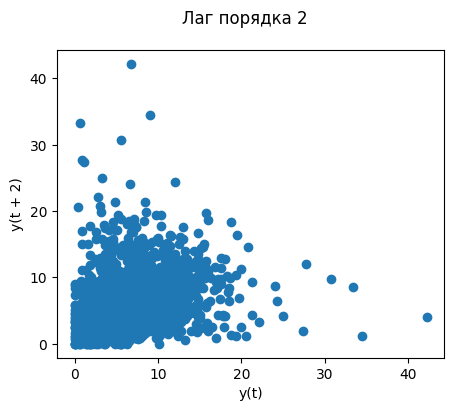
date  
2013-01-01 0.000000  
2013-01-02 2.980000  
2013-01-03 4.633333  
2013-01-04 1.233333  
2013-01-05 3.700000  
Name: wind\_speed, dtype: float64

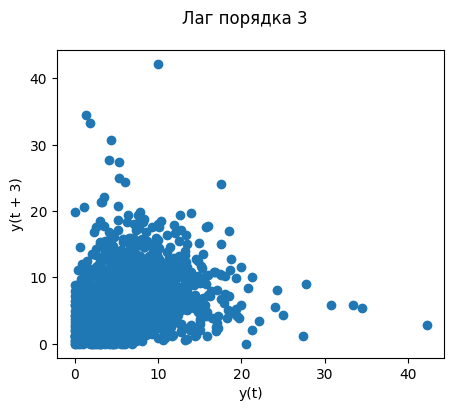
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))  
fig.suptitle('Временной ряд в виде графика')  
data\_train.plot(ax=ax, legend=False)  
pyplot.show()



for i in range(1, 5):  
 fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(5,4))  
 fig.suptitle(f'Лаг порядка {i}')  
 pd.plotting.lag\_plot(data\_train, lag=i, ax=ax)  
 pyplot.show()

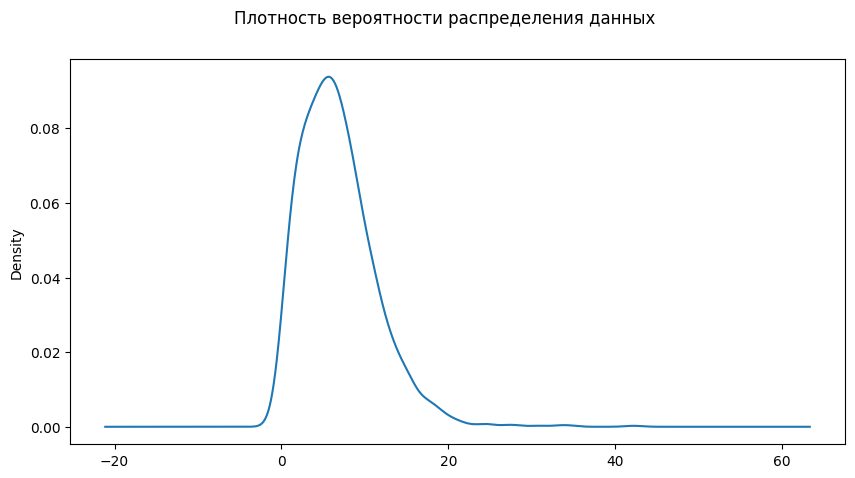




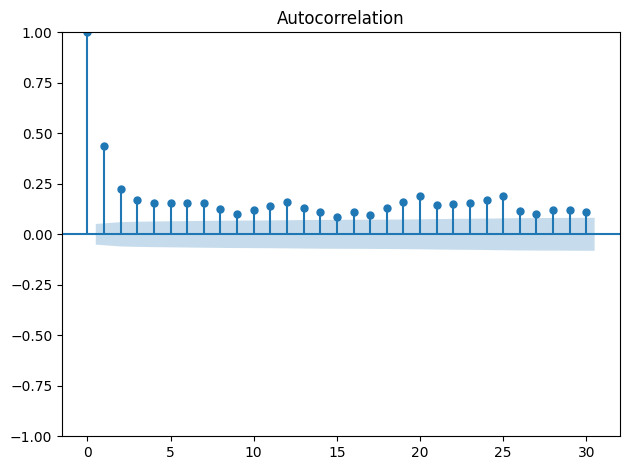




fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))  
fig.suptitle('Плотность вероятности распределения данных')  
data\_train.plot(ax=ax, kind='kde', legend=False)  
pyplot.show()

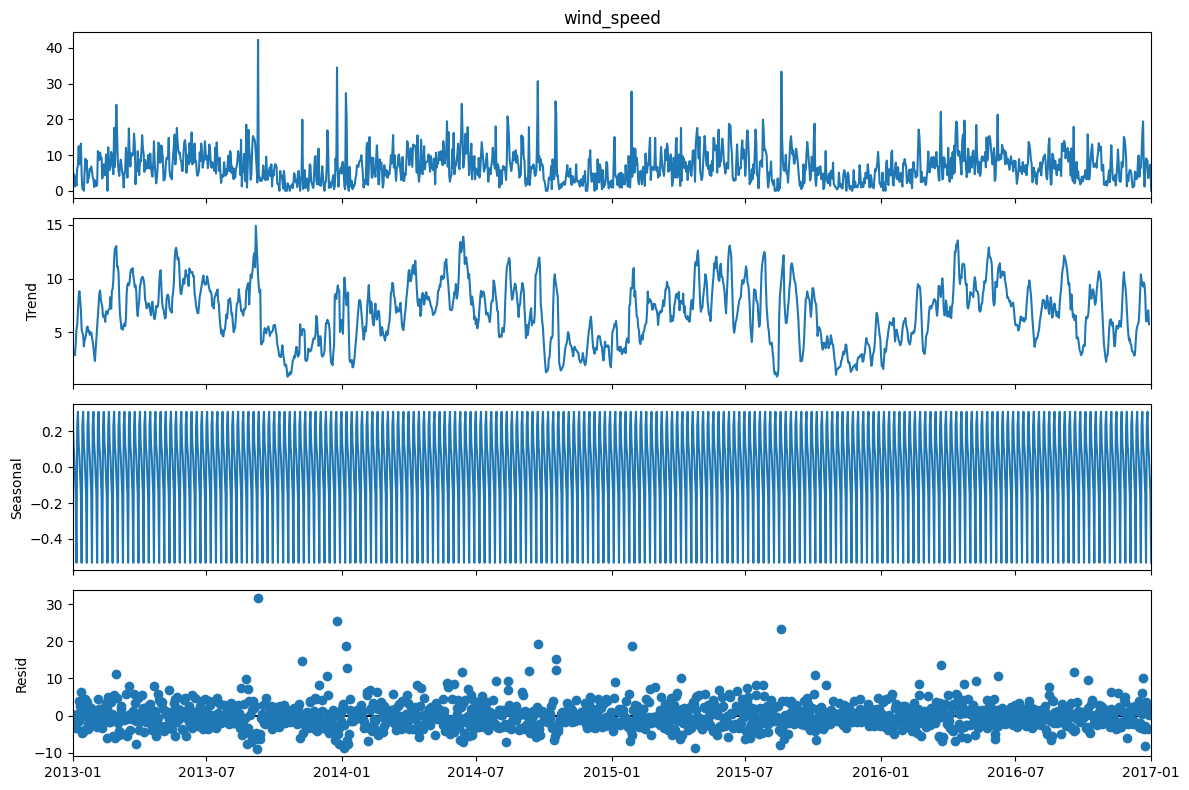


from statsmodels.graphics.tsaplots import plot\_acf  
plot\_acf(data\_train, lags=30)  
plt.tight\_layout()



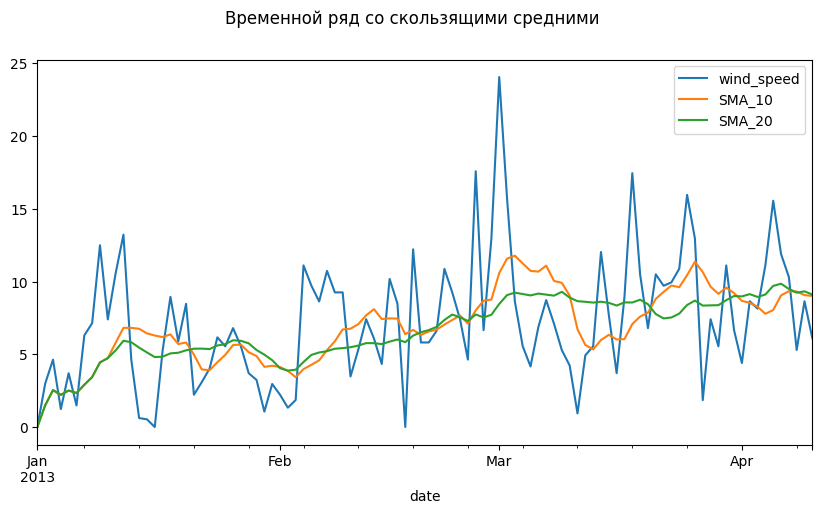
# https://www.statsmodels.org/dev/generated/statsmodels.tsa.seasonal.seasonal\_decompose.html  
from statsmodels.tsa.seasonal import seasonal\_decompose  
# Аддитивная модель  
def plot\_decompose(data=data\_train, model='add'):  
 result\_add = seasonal\_decompose(data, model = 'add')  
 fig = result\_add.plot()  
 fig.set\_size\_inches((12, 8))  
 # Перерисовка  
 fig.tight\_layout()  
 plt.show()

plot\_decompose()



data\_train2 = data\_start.copy()  
  
# Простое скользящее среднее (SMA)  
data\_train2['SMA\_10'] = data\_train2['wind\_speed'].rolling(10, min\_periods=1).mean()  
data\_train2['SMA\_20'] = data\_train2['wind\_speed'].rolling(20, min\_periods=1).mean()  
  
print(data\_train2)  
  
fig, ax = pyplot.subplots(1, 1, sharex='col', sharey='row', figsize=(10,5))  
fig.suptitle('Временной ряд со скользящими средними')  
data\_train2[:100].plot(ax=ax, legend=True)  
pyplot.show()

meantemp humidity wind\_speed meanpressure SMA\_10 SMA\_20  
date   
2013-01-01 None None 0.000000 None 0.000000 0.000000  
2013-01-02 None None 2.980000 None 1.490000 1.490000  
2013-01-03 None None 4.633333 None 2.537778 2.537778  
2013-01-04 None None 1.233333 None 2.211667 2.211667  
2013-01-05 None None 3.700000 None 2.509333 2.509333  
... ... ... ... ... ... ...  
2016-12-28 None None 3.547826 None 8.812007 7.059632  
2016-12-29 None None 6.000000 None 8.370578 7.294007  
2016-12-30 None None 6.266667 None 7.404578 7.339562  
2016-12-31 None None 7.325000 None 6.196602 7.469979  
2017-01-01 None None 0.000000 None 5.578507 7.305888  
  
[1462 rows x 6 columns]



from gplearn.genetic import SymbolicRegressor  
xnum = list(range(data\_train.shape[0]))  
print(data\_train)  
  
function\_set = ['add', 'sub', 'mul', 'div', 'sin']  
est\_gp = SymbolicRegressor(population\_size=2000, metric='mse',  
 generations=40, stopping\_criteria=0.01,  
 init\_depth=(5, 10), verbose=1, function\_set=function\_set,  
 const\_range=(-100, 100), random\_state=0)

date  
2013-01-01 0.000000  
2013-01-02 2.980000  
2013-01-03 4.633333  
2013-01-04 1.233333  
2013-01-05 3.700000  
 ...   
2016-12-28 3.547826  
2016-12-29 6.000000  
2016-12-30 6.266667  
2016-12-31 7.325000  
2017-01-01 0.000000  
Name: wind\_speed, Length: 1462, dtype: float64

est\_gp.fit(np.array(xnum).reshape(-1, 1), data\_train.values.reshape(-1, 1))

C:\Users\prite\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9\_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\sklearn\utils\validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().  
 y = column\_or\_1d(y, warn=True)

| Population Average | Best Individual |  
---- ------------------------- ------------------------------------------ ----------  
 Gen Length Fitness Length Fitness OOB Fitness Time Left  
 0 297.43 9.43328e+84 39 28.775 N/A 12.86m  
 1 143.69 6.80794e+24 1 21.19 N/A 4.81m  
 2 166.51 1.14107e+28 31 21.1583 N/A 4.89m  
 3 61.78 1.19262e+30 54 20.7418 N/A 2.44m  
 4 66.11 3.8987e+27 49 20.7036 N/A 2.45m  
 5 48.95 1.40798e+18 89 20.3482 N/A 2.31m  
 6 45.22 7.09192e+18 94 20.3169 N/A 2.12m  
 7 53.20 5.62913e+09 94 20.1546 N/A 2.48m  
 8 74.17 5.39694e+12 94 20.0701 N/A 2.45m  
 9 96.85 1.53548e+09 94 20.0701 N/A 3.23m  
 10 100.87 2.33378e+09 95 19.6012 N/A 2.71m  
 11 105.54 1.22694e+09 95 19.6012 N/A 3.00m  
 12 107.11 6.42066e+07 92 19.5817 N/A 2.64m  
 13 109.15 1.54232e+10 101 19.5498 N/A 2.89m  
 14 108.67 3.11581e+08 197 19.353 N/A 2.64m  
 15 102.74 5.36926e+20 197 19.348 N/A 2.60m  
 16 111.07 2.47606e+11 166 19.095 N/A 2.78m  
 17 159.09 1.5796e+08 176 19.0736 N/A 3.32m  
 18 180.97 6.10213e+08 334 18.9785 N/A 3.45m  
 19 162.14 8.20035e+06 347 18.901 N/A 3.00m  
 20 170.77 6.43466e+08 174 18.571 N/A 2.88m  
 21 174.03 3.83843e+10 173 18.555 N/A 2.75m  
 22 177.52 1.52191e+14 160 18.4278 N/A 2.81m  
 23 189.01 2.33866e+08 253 18.1846 N/A 2.81m  
 24 180.82 5.1718e+06 339 18.1173 N/A 2.42m  
 25 182.46 3.32823e+08 228 18.0277 N/A 2.18m  
 26 203.65 1.25802e+11 258 17.845 N/A 2.23m  
 27 231.16 3.07297e+08 285 17.772 N/A 2.37m  
 28 218.41 2.70998e+06 296 17.7681 N/A 2.10m  
 29 218.16 3.05825e+08 414 17.6717 N/A 1.98m  
 30 246.08 1.26852e+09 282 17.6014 N/A 1.95m  
 31 246.72 1.1634e+06 412 17.5028 N/A 1.80m  
 32 238.89 1.24001e+06 412 17.5028 N/A 1.41m  
 33 236.99 2.59995e+06 506 17.4542 N/A 1.18m  
 34 233.55 690935 287 17.4636 N/A 1.04m  
 35 219.81 7.32465e+06 259 17.4514 N/A 43.18s  
 36 209.18 1.04928e+14 267 17.4386 N/A 34.33s  
 37 209.59 5.16712e+06 284 17.4153 N/A 22.15s  
 38 220.14 1.84657e+06 275 17.3796 N/A 12.33s  
 39 224.67 6.65885e+08 304 17.3242 N/A 0.00s

SymbolicRegressor(const\_range=(-100, 100),  
 function\_set=['add', 'sub', 'mul', 'div', 'sin'],  
 generations=40, init\_depth=(5, 10), metric='mse',  
 population\_size=3000, random\_state=0, stopping\_criteria=0.01,  
 verbose=1)

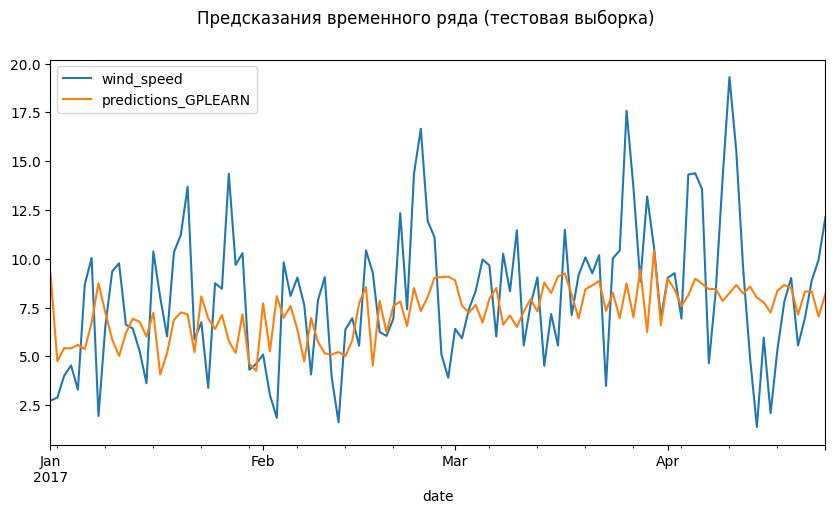
import graphviz  
dot\_data = est\_gp.\_program.export\_graphviz()  
graph = graphviz.Source(dot\_data)  
graph

xnum\_test = list(range(data\_test.shape[0]))  
# Предсказания  
y\_gp = est\_gp.predict(np.array(xnum\_test).reshape(-1, 1))  
y\_gp[:10]  
  
mean\_squared\_error(test, y\_gp, squared=False)

3.6427624900222955

# Записываем предсказания в DataFrame  
data\_test['predictions\_GPLEARN'] = list(y\_gp)

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



# Целочисленная метка шкалы времени   
xnum = list(range(data\_start.shape[0] + data\_test.shape[0]))  
# Разделение выборки на обучающую и тестовую  
train\_size = len(data\_start)  
train, test = data\_start['wind\_speed'].values, data\_test['wind\_speed'].values  
history\_arima = [x for x in train]  
history\_es = [x for x in train]  
  
print(train)  
print(test)

[0. 2.98 4.63333333 ... 6.26666667 7.325 0. ]  
[ 2.74347826 2.89444444 4.01666667 4.545 3.3 8.68181818  
 10.04166667 1.95 6.54285714 9.36111111 9.77222222 6.62631579  
 6.43529412 5.276 3.63043478 10.38 8.03888889 6.02916667  
 10.33809524 11.22631579 13.69565217 5.868 6.75294118 3.39130435  
 8.756 8.46785714 14.35833333 9.69090909 10.29444444 4.32222222  
 4.625 5.1 3.02727273 1.85454545 9.82 8.1  
 9.04444444 7.6375 4.08 7.875 9.06666667 3.95  
 1.625 6.37777778 6.9625 5.55714286 10.4375 9.28  
 6.25 6.05454545 6.9375 12.34166667 7.4125 14.35  
 16.6625 11.92857143 11.1125 5.1 3.91111111 6.41538462  
 5.93 7.4125 8.35 9.9625 9.66666667 6.025  
 10.26363636 8.34285714 11.4625 5.56666667 7.6375 9.05555556  
 4.52222222 7.175 5.56 11.49 7.12307692 9.16111111  
 10.07777778 9.2625 10.1875 3.4875 10.03333333 10.425  
 17.59 13.65 8.84444444 13.2 10.58571429 6.95  
 9.0375 9.2625 6.9375 14.32 14.38461538 13.57777778  
 4.65 8.3375 14.125 19.31428571 15.5125 9.4875  
 4.94444444 1.3875 5.96666667 2.1 5.36666667 7.81111111  
 9.025 5.5625 6.9625 8.89 9.9625 12.15714286]

from sklearn.metrics import mean\_squared\_error  
from statsmodels.tsa.arima.model import ARIMA  
from statsmodels.tsa.holtwinters import ExponentialSmoothing  
  
# Параметры модели (p,d,q)   
arima\_order = (6,1,0)  
# Формирование предсказаний  
predictions\_arima = list()  
for t in range(len(test)):  
 model\_arima = ARIMA(history\_arima, order=arima\_order)  
 model\_arima\_fit = model\_arima.fit()  
 yhat\_arima = model\_arima\_fit.forecast()[0]  
 predictions\_arima.append(yhat\_arima)  
 history\_arima.append(test[t])  
# Вычисление метрики RMSE

mean\_squared\_error(test, predictions\_arima, squared=False)

3.3452863559955994

data\_test['predictions\_ARIMA'] = list(predictions\_arima)

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

