

МГТУ им. Н. Э. Баумана, кафедра ИУ5
курс “Технология машинного обучения”

Курсовая работа
**«Проектирование модели машинного
обучения»**

ВЫПОЛНИЛ:

Матюнин да Вейга Р.А.

Группа: ИУ5-61Б

ПРОВЕРИЛ:

Гапанюк Ю.Е.

Москва 2020

Содержание

Оглавление

Содержание	2
Введение	3
Подготовка	4
Постановка задачи.....	6
Разведочный анализ	8
Основная часть	11
Заключение	19
Список литературы	20

Введение

Курсовой проект – самостоятельная часть учебной дисциплины «Технологии машинного обучения» – учебная и практическая исследовательская студенческая работа, направленная на решение комплексной задачи машинного обучения. Результатом курсового проекта является отчет, содержащий описания моделей, тексты программ и результаты экспериментов.

Курсовой проект опирается на знания, умения и владения, полученные студентом в рамках лекций и лабораторных работ по дисциплине.






Целью данной работы является, исследование и анализ данных, представленных в текстовых файлах (формата csv), для дальнейшего построения модели машинного обучения в прогнозировании климатических изменений.

Подготовка

В качестве датасета для данной работы, я выбрал данные о климатических изменениях на поверхности Земли, собранные и предоставленные на сайте:

- <https://www.kaggle.com/berkeleyearth/climate-change-earth-surface-temperature-data/data>

Датасет представляет из себя набор данных мировых температур от 1750 до 2015 гг. и содержит в себе несколько файлов:

 GlobalLandTemperaturesByCity.csv	19.09.2019 4:47	Файл Microsoft E...	520 343 КБ
 GlobalLandTemperaturesByCountry.csv	19.09.2019 4:47	Файл Microsoft E...	22 149 КБ
 GlobalLandTemperaturesByMajorCity.csv	19.09.2019 4:47	Файл Microsoft E...	13 808 КБ
 GlobalLandTemperaturesByState.csv	19.09.2019 4:47	Файл Microsoft E...	30 049 КБ
 GlobalTemperatures.csv	08.02.2020 12:25	Файл Microsoft E...	202 КБ

GlobalTemperaturesByCity.csv содержит следующие поля:

- **Dt:** поле даты, которая содержит месяц и год.
- **AverageTemperature:** средняя температура.
- **AverageTemperatureUncertainty:** 95% доверительный интервал от среднего значения температуры.
- **City:** город.
- **Country:** страна.
- **Latitude:** долгота.
- **Longitude:** широта.

GlobalTemperaturesByCountry.csv содержит следующие поля:

- **Dt:** поле даты, которая содержит месяц и год.
- **AverageTemperature:** средняя температура.
- **AverageTemperatureUncertainty:** 95% доверительный интервал от среднего значения температуры.
- **Country:** страна.

GlobalTemperaturesByMajorCity.csv содержит следующие поля:

- **Dt:** поле даты, которая содержит месяц и год.
- **AverageTemperature:** средняя температура.
- **AverageTemperatureUncertainty:** 95% доверительный интервал от среднего значения температуры.
- **City:** город.
- **Country:** страна.
- **Latitude:** долгота.
- **Longitude:** широта.

GlobalTemperaturesByState.csv содержит следующие поля:

- **Dt:** поле даты, которая содержит месяц и год.
- **AverageTemperature:** средняя температура.
- **AverageTemperatureUncertainty:** 95% доверительный интервал от среднего значения температуры.
- **State:** область.
- **Country:** страна.

GlobalTemperatures.csv содержит следующие поля:

- **Date:** поле даты, которая содержит месяц и год. Поля 1750-1850 гг. содержат информацию только о средних температурах (**LandAverageTemperature** и **LandAverageTemperatureUncertainty**), 1850-2015 гг. еще и о максимальной и минимальной температуры земли и мировой температуры океана и земли.
- **LandAverageTemperature:** средняя температура Земли (в Цельсиях).
- **LandAverageTemperatureUncertainty:** 95% доверительный интервал от среднего значения температуры
- **LandMaxTemperature:** максимальная температура Земли (в Цельсиях)
- **LandMaxTemperatureUncertainty:** 95% доверительный интервал от максимального значения температуры
- **LandMinTemperature:** минимальная температура Земли (в Цельсиях)
- **LandMinTemperatureUncertainty:** 95% доверительный интервал от минимального значения температуры
- **LandAndOceanAverageTemperature:** средняя температура суши и океана (в Цельсиях)
- **LandAndOceanAverageTemperatureUncertainty:** 95% доверительный интервал от среднего значения температуры суши и океана/

Постановка задачи

Климатические изменения - всем известный факт. С каждым годом мы замечаем, какие последствия несут за собой эти изменения. С современными технологиями мы можем быть готовыми встретить эти изменения и свести потери к минимуму.

Давайте наглядно рассмотрим данную проблему.

```
In [1]: import pandas as pd
import seaborn as sns
import numpy as np

import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

```
In [2]: global_temp = pd.read_csv('data/GlobalTemperatures.csv')
```

```
In [3]: global_temp = global_temp[['dt', 'LandAverageTemperature']]

global_temp['dt'] = pd.to_datetime(global_temp['dt'])
global_temp['year'] = global_temp['dt'].map(lambda x: x.year)
global_temp['month'] = global_temp['dt'].map(lambda x: x.month)

def get_season(month):
    if month >= 3 and month <= 5:
        return 'spring'
    elif month >= 6 and month <= 8:
        return 'summer'
    elif month >= 9 and month <= 11:
        return 'autumn'
    else:
        return 'winter'

min_year = global_temp['year'].min()
max_year = global_temp['year'].max()
years = range(min_year, max_year + 1)

global_temp['season'] = global_temp['month'].apply(get_season)

spring_temps = []
summer_temps = []
autumn_temps = []
winter_temps = []

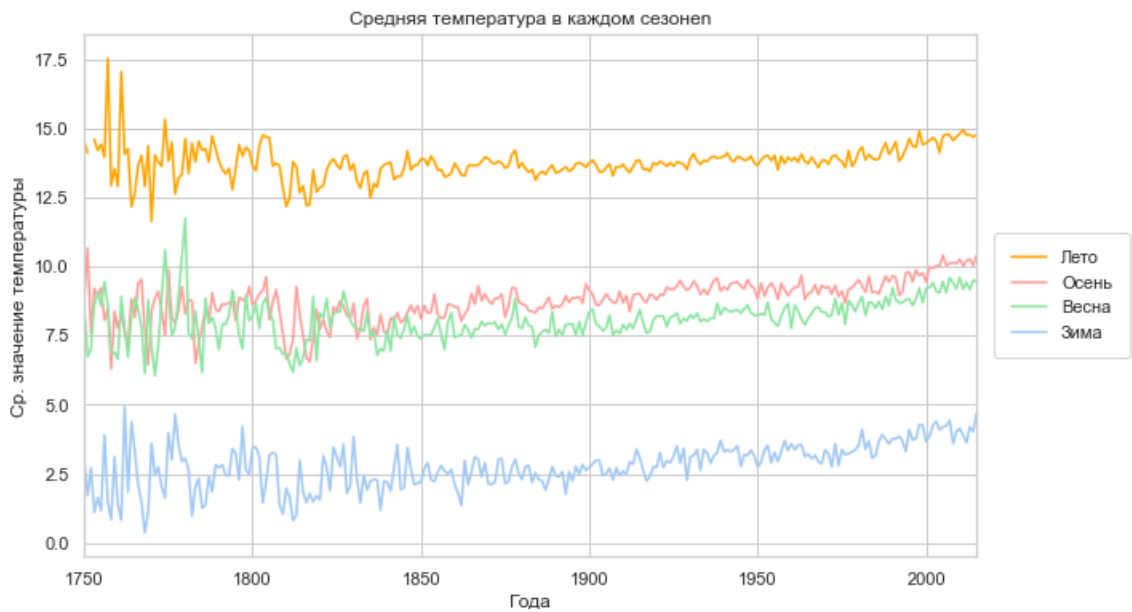
for year in years:
    curr_years_data = global_temp[global_temp['year'] == year]
    spring_temps.append(curr_years_data[curr_years_data['season'] == 'spring']['LandAverageTemperature'])
    summer_temps.append(curr_years_data[curr_years_data['season'] == 'summer']['LandAverageTemperature'])
    autumn_temps.append(curr_years_data[curr_years_data['season'] == 'autumn']['LandAverageTemperature'])
    winter_temps.append(curr_years_data[curr_years_data['season'] == 'winter']['LandAverageTemperature'])
```

```
In [4]: sns.set(style="whitegrid")
sns.set_color_codes("pastel")
f, ax = plt.subplots(figsize=(10, 6))

plt.plot(years, summer_temps, label='Лето', color='orange')
plt.plot(years, autumn_temps, label='Осень', color='r')
plt.plot(years, spring_temps, label='Весна', color='g')
plt.plot(years, winter_temps, label='Зима', color='b')

plt.xlim(min_year, max_year)

ax.set_ylabel('Ср. значение температуры')
ax.set_xlabel('Года')
ax.set_title('Средняя температура в каждом сезоне')
legend = plt.legend(loc='center left', bbox_to_anchor=(1, 0.5), frameon=True, borderpad=1, borderaxes)
< >
```



Как видно из графиков (если учесть возможность недостоверности данных 18-19 вв.) можно заметить рост температуры более чем на 3 градуса, особенно зимой.

Давайте рассмотрим данную проблему более подробно.

Разведочный анализ

Подключим нужные библиотеки.

```
In [1]: import pandas as pd
import seaborn as sns
import numpy as np

import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from sklearn.metrics import mean_squared_error
from math import sqrt
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

Просмотрим файл GlobalLandTemperaturesByCity.csv:

```
In [7]: cities = pd.read_csv('data/GlobalLandTemperaturesByCity.csv', sep=",")
```

```
In [8]: cities.head()
```

```
Out[8]:
```

	dt	AverageTemperature	AverageTemperatureUncertainty	City	Country	Latitude	Longitude
0	1743-11-01	6.068	1.737	Århus	Denmark	57.05N	10.33E
1	1743-12-01	NaN	NaN	Århus	Denmark	57.05N	10.33E
2	1744-01-01	NaN	NaN	Århus	Denmark	57.05N	10.33E
3	1744-02-01	NaN	NaN	Århus	Denmark	57.05N	10.33E
4	1744-03-01	NaN	NaN	Århus	Denmark	57.05N	10.33E

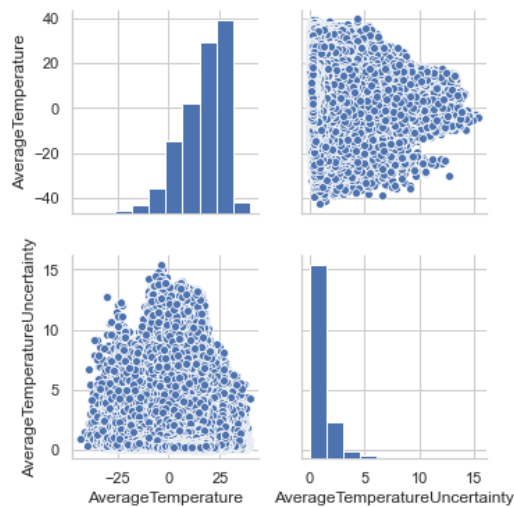
```
In [9]: cities.isnull().sum()
```

```
Out[9]: dt                0
AverageTemperature      364130
AverageTemperatureUncertainty  364130
City                    0
Country                 0
Latitude                0
Longitude               0
dtype: int64
```



```
In [19]: sns.pairplot(cities)
```

```
Out[19]: <seaborn.axisgrid.PairGrid at 0x2e91e1a9d90>
```



Изменим наш датасет преобразовав часть файла, связанного только с Москвой:

```
In [7]: cities = pd.read_csv('data/GlobalLandTemperaturesByCity.csv')
mos = cities.loc[cities['City'] == 'Moscow', ['dt', 'AverageTemperature']]
mos.columns = ['Date', 'Temp']
mos['Date'] = pd.to_datetime(mos['Date'])
mos.reset_index(drop=True, inplace=True)
mos.set_index('Date', inplace=True)

mos = mos.loc['1900': '2013-01-01']
mos = mos.asfreq('M', method='bfill')
mos.head()
```

```
Out[7]:
```

Date	Temp
1900-01-31	-10.109
1900-02-28	-4.419
1900-03-31	2.300
1900-04-30	9.853
1900-05-31	13.814

```
In [8]: mos.shape
```

```
Out[8]: (1356, 1)
```

```
In [9]: mos.columns
```

```
Out[9]: Index(['Temp'], dtype='object')
```

```
In [10]: mos.dtypes
```

```
Out[10]: Temp    float64
dtype: object
```

```
In [11]: mos.isnull().sum()
```

```
Out[11]: Temp    0
dtype: int64
```

Представленный набор данных уже не содержит пропусков

Т.к. представленный датасет весьма специфический, обычные модели для решения задач классификации или регрессии не подойдут.

Т.к. существуют некоторые переменные, которые зависят от времени, к примеру, некоторые величины могут иметь эффективное отношение к величинам, которые произошли в прошлом. В настоящее время существует несколько типов моделей прогнозирования временных рядов, поэтому для данной работы я решил использовать сезонные модели ARIMA.

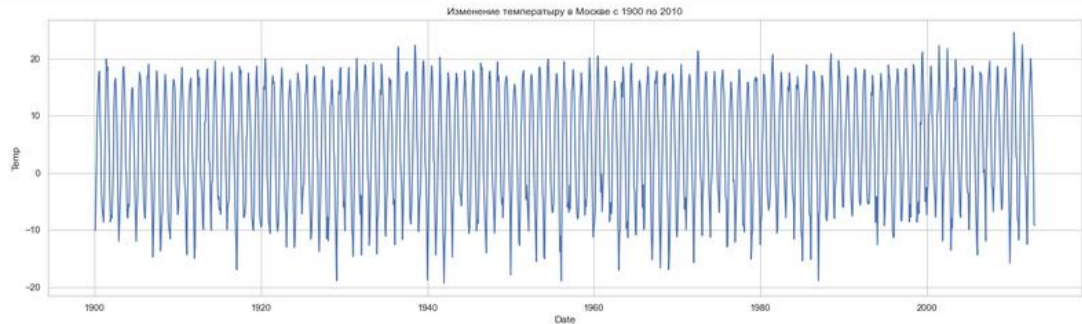
SARIMA расшифровывается как Seasonal Auto Regressive Integrated Moving Average - Сезонная Авто Регрессивная Интегрированная Скользящая Средняя.

В статистике и эконометрике, в частности в анализе временных рядов, модель авторегрессионного интегрированного скользящего среднего (ARIMA) является обобщением модели авторегрессионного скользящего среднего (ARMA). Обе эти модели адаптированы к данным временного ряда либо для лучшего понимания данных, либо для прогнозирования будущих точек в ряду (прогнозирование).

Основная часть

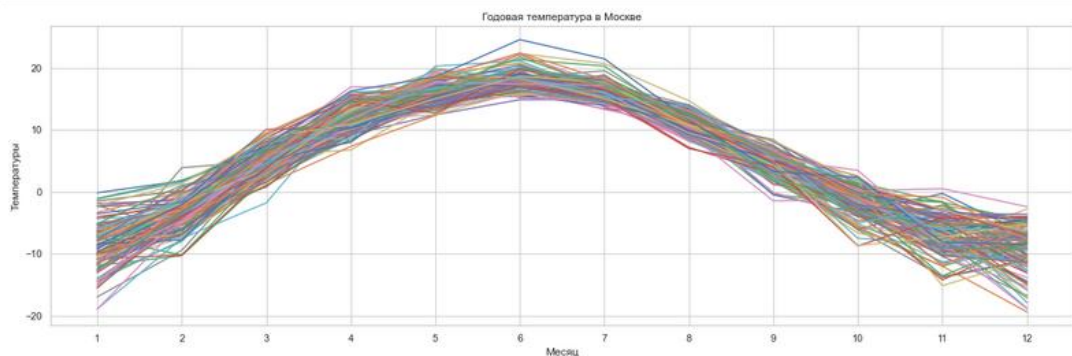
Посмотрим какие изменения температуры произошли в Москве с период с 1900 по 2010:

```
In [17]: plt.figure(figsize=(22,6))
sns.lineplot(x=moc.index, y=moc['Temp'])
plt.title('Изменение температуры в Москве с 1900 по 2010')
plt.show()
```



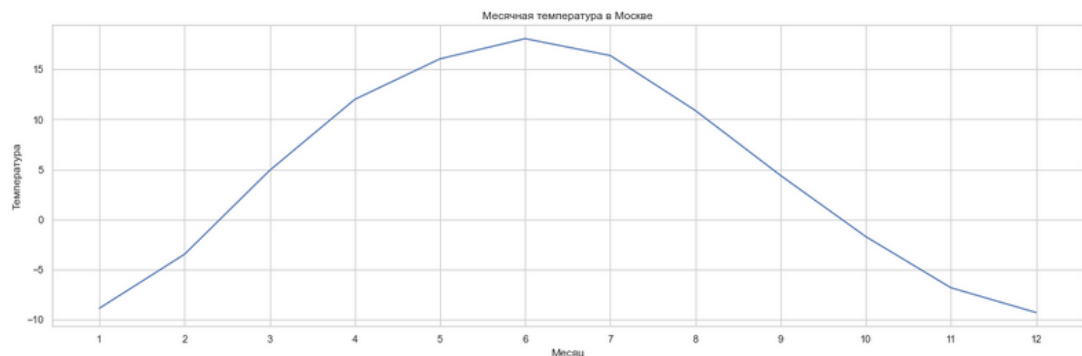
Можно так же заметить, что ряд явно имеет некоторую сезонность: более высокие температуры около июня и июля, и более низкие в декабре и январе.

```
In [18]: moc['month'] = moc.index.month
moc['year'] = moc.index.year
pivot = pd.pivot_table(moc, values='Temp', index='month', columns='year', aggfunc='mean')
pivot.plot(figsize=(20,6))
plt.title('Годовая температура в Москве')
plt.xlabel('Месяц')
plt.ylabel('Температуры')
plt.xticks([x for x in range(1,13)])
plt.legend().remove()
plt.show()
```



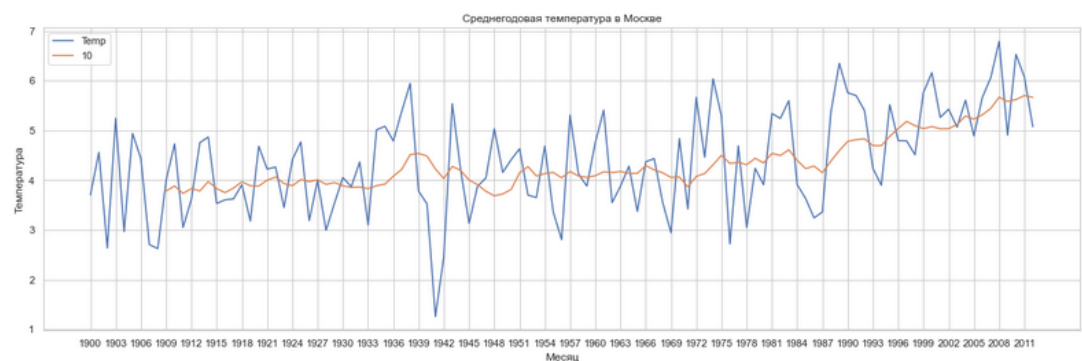
Давайте объединим все это в один график:

```
In [19]: monthly_seasonality = pivot.mean(axis=1)
monthly_seasonality.plot(figsize=(20,6))
plt.title('Месячная температура в Москве')
plt.xlabel('Месяц')
plt.ylabel('Температура')
plt.xticks([x for x in range(1,13)])
plt.show()
```



Теперь проверим, существует ли какая-либо тенденция:

```
In [20]: year_avg = pd.pivot_table(mos, values='Temp', index='year', aggfunc='mean')
year_avg['10'] = year_avg['Temp'].rolling(10).mean()
year_avg[['Temp', '10']].plot(figsize=(20,6))
plt.title('Среднегодовая температура в Москве')
plt.xlabel('Месяц')
plt.ylabel('Температура')
plt.xticks([x for x in range(1900,2012,3)])
plt.show()
```



И как видно она существует. Средняя температура увеличилась с $4,8^{\circ}$ до $5,7^{\circ}$, то есть примерно 4,2% за 100 лет.

Теперь прежде, чем создать прогноз, создадим базовый прогноз в наборе проверки. Модель будет рассматривать предыдущий месяц в качестве базового прогноза на следующий месяц:

```

In [21]: train = mos[:-60].copy()
         val = mos[-60:-12].copy()
         test = mos[-12:].copy()

In [22]: baseline = val['Temp'].shift()
         baseline.dropna(inplace=True)
         baseline.head()

Out[22]: Date
2008-02-29    -2.197
2008-03-31     1.340
2008-04-30     8.806
2008-05-31    10.868
2008-06-30    15.058
Freq: M, Name: Temp, dtype: float64

In [23]: def measure_rmse(y_true, y_pred):
         return sqrt(mean_squared_error(y_true, y_pred))

rmse_base = measure_rmse(val.iloc[1:,0],baseline)
print(f'RMSE baseline, которую мы будем пытаться уменьшить, составляет {round(rmse_base,4)} градус Це

<
RMSE базовой линии, которую мы будем пытаться уменьшить, составляет 5.9605 градус Цельсия

```

Как мы видим, ряд имеет небольшой восходящую тенденцию, и кажется, что существует некоторая сезонность с более низкими температурами в начале и в конце года и более высокими температурами в середине года.

Для создания прогноза временного ряда ряд должен быть стационарным (постоянное среднее, дисперсия и автокорреляция).

Один из способов проверить, является ли ряд стационарным, заключается в использовании функции `adfuller`. Если значение P меньше 5% (обычное число, используемое для такого рода исследований), серия является стационарной, и вы можете начать создавать свою модель.

Ниже приведена функция проверки стационарности, она показывает:

- Сам ряд.
- Автокорреляционная функция (АКФ):
 - Она показывает корреляцию между текущими температурами и запаздывающими версиями себя.
- Частичная автокорреляция (ПАСФ):
 - Она показывает корреляцию между текущими температурами и лаговой версией, исключая эффекты более ранних лагов, например, он показывает эффективное влияние лага 3 на текущие температуры, исключая эффекты лагов 1 и 2.

```
In [24]: def check_stationarity(y, lags_plots=48, figsize=(22,8)):
    "Use Series as parameter"

    # Creating plots of the DF
    y = pd.Series(y)
    fig = plt.figure()

    ax1 = plt.subplot2grid((3, 3), (0, 0), colspan=2)
    ax2 = plt.subplot2grid((3, 3), (1, 0))
    ax3 = plt.subplot2grid((3, 3), (1, 1))
    ax4 = plt.subplot2grid((3, 3), (2, 0), colspan=2)

    y.plot(ax=ax1, figsize=figsize)
    ax1.set_title('Изменения температуры в Москве')
    plot_acf(y, lags=lags_plots, zero=False, ax=ax2);
    plot_pacf(y, lags=lags_plots, zero=False, ax=ax3);
    sns.distplot(y, bins=int(sqrt(len(y))), ax=ax4)
    ax4.set_title('Диаграмма распределения')

    plt.tight_layout()

    print('Результаты теста:')
    adfinput = adfuller(y)
    adftest = pd.Series(adfinput[0:4], index=['Статистика теста', 'p-величина', 'Использовано лагов', ''])
    adftest = round(adftest,4)

    for key, value in adfinput[4].items():
        adftest["Критическое значение (%s)"%key] = value.round(4)

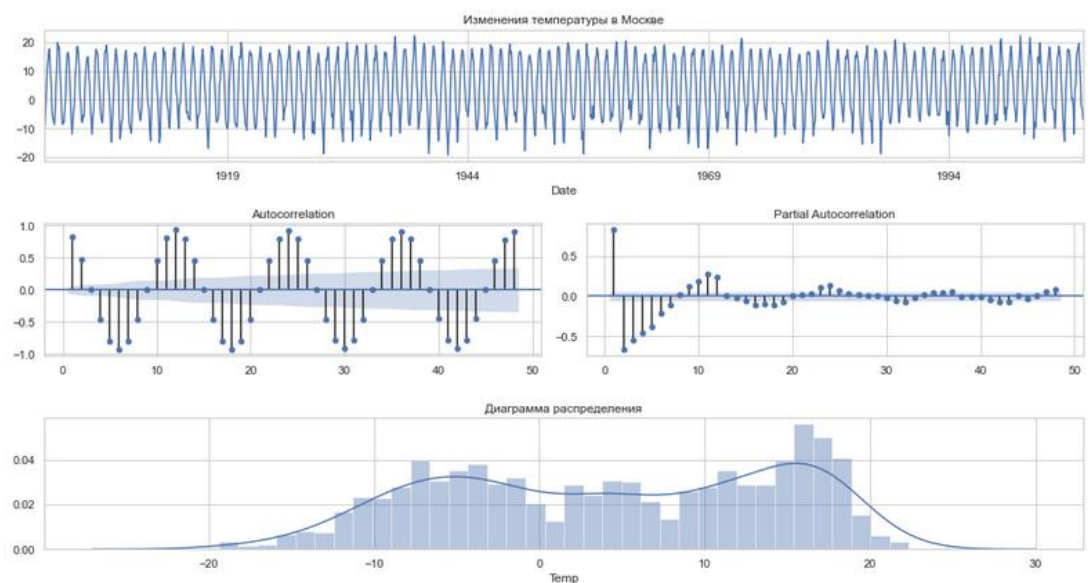
    print(adftest)

    if adftest[0].round(2) < adftest[5].round(2):
        print('\nСтатистика теста ниже критического значения 5%.\nРяд стационарна')
    else:
        print("\nСтатистика теста выше критического значения 5%.\nРяд не стационарна")
```

```
In [25]: check_stationarity(train['Temp'])
```

```
Результаты теста:
Статистика теста          -5.4123
p-величина                0.0000
Использовано лагов        23.0000
Количество использованных наблюдений  1272.0000
Критическое значение (1%)   -3.4355
Критическое значение (5%)  -2.8638
Критическое значение (10%) -2.5680
dtype: float64
```

```
Статистика теста ниже критического значения 5%.
Ряд стационарна
```



Ряд имеет интересное поведение, есть последовательная значительная отрицательная автокорреляция, повторяющаяся каждые 12 месяцев, это из-за разницы в сезонах, если сегодня зима с холодными температурами через 6 месяцев, у нас будут более высокие температуры в летом, поэтому возникает отрицательная автокорреляция. Эти температуры обычно идут в противоположных направлениях.

PACF показывает положительный всплеск в первом запаздывании и снижение к отрицательному PACF в следующих запаздываниях.

Такое поведение между графиками ACF и PACF предполагает модель AR (1), а также первую сезонную разницу. Я снова нанесу на график функцию стационарности с первой сезонной разницей, чтобы узнать, понадобится ли нам какой-нибудь параметр SAR (P) или SMA (Q):

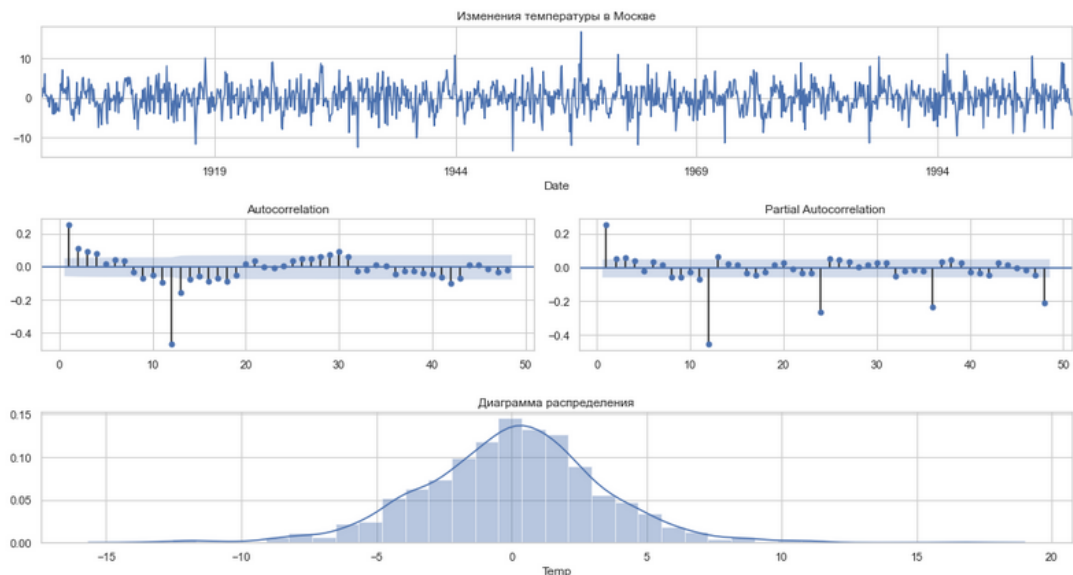
```
In [26]: check_stationarity(train['Temp'].diff(12).dropna())
```

Результаты теста:

Статистика теста	-12.1908
p-величина	0.0000
Использовано лагов	23.0000
Количество использованных наблюдений	1260.0000
Критическое значение (1%)	-3.4356
Критическое значение (5%)	-2.8638
Критическое значение (10%)	-2.5680
dtype:	float64

Статистика теста ниже критического значения 5%.

Ряд стационарна



Как показано на графиках выше, первые лаги ACF имеют постепенное затухание, в то время как PACF падает ниже доверительного интервала после третьего лага, это AR-сигнатура с параметром 3.

Чтобы начать прогнозирование набора проверки, я собираюсь создать функцию для использования одношагового прогноза во всем наборе проверки и измерения ошибки:

```
In [27]: def walk_forward(training_set, validation_set, params):
        """
        Параметры: это кортеж, в котором мы собрали следующие параметры SARIMA: ((pdq), (PDQS), trend)
        """
        history = [x for x in training_set.values]
        prediction = list()

        pdq, PDQS, trend = params

        for week in range(len(validation_set)):
            model = sm.tsa.statespace.SARIMAX(history, order=pdq, seasonal_order=PDQS, trend=trend)
            result = model.fit(dispatch=False)
            yhat = result.predict(start=len(history), end=len(history))
            prediction.append(yhat[0])
            history.append(validation_set[week])

        return prediction

In [28]: val['Pred'] = walk_forward(train['Temp'], val['Temp'], ((3,0,0),(0,1,1,12),'c'))

In [29]: rmse_pred = measure_rmse(val['Temp'], val['Pred'])

print(f"Среднеквадратичное отклонение модели SARIMA(3,0,0),(0,1,1,12),'c' составило {round(rmse_pred, 2)}")
print(f"Это снижение на среднеквадратичное отклонение равно {round((rmse_pred/rmse_base-1)*100,2)}%")

<----->

Среднеквадратичное отклонение модели SARIMA(3,0,0),(0,1,1,12),'c' составило 2.4881 градуса Цельсия
Это снижение на среднеквадратичное отклонение равно -58.26%

In [30]: # Создание столбца ошибок
val['Error'] = val['Temp'] - val['Pred']
```

Всегда важно проверять остатки, я собираюсь создать функцию для построения некоторых важных диаграмм, чтобы помочь нам визуализировать остатки.

Я собираюсь построить следующие графики:

- Текущие и прогнозируемые значения за время.
- Остаточные и прогнозные значения в диаграмме рассеяния.
- График, показывающий распределение ошибок и его идеальное распределение
- График автокорреляции остатков, чтобы увидеть, есть ли еще какая-то корреляция.


```
In [31]: def plot_error(data, figsize=(20,8)):
'''
Сделаем 3 столбца: Temperature, Prediction, Error
'''
plt.figure(figsize=figsize)
ax1 = plt.subplot2grid((2,2), (0,0))
ax2 = plt.subplot2grid((2,2), (0,1))
ax3 = plt.subplot2grid((2,2), (1,0))
ax4 = plt.subplot2grid((2,2), (1,1))

ax1.plot(data.iloc[:,0:2])
ax1.legend(['Real', 'Pred'])
ax1.set_title('Current and Predicted Values')

ax2.scatter(data.iloc[:,1], data.iloc[:,2])
ax2.set_xlabel('Предсказанные значения')
ax2.set_ylabel('Ошибки')
ax2.set_title('Отношение ошибок к предсказанным величинам')

sm.graphics.qqplot(data.iloc[:,2], line='r', ax=ax3)

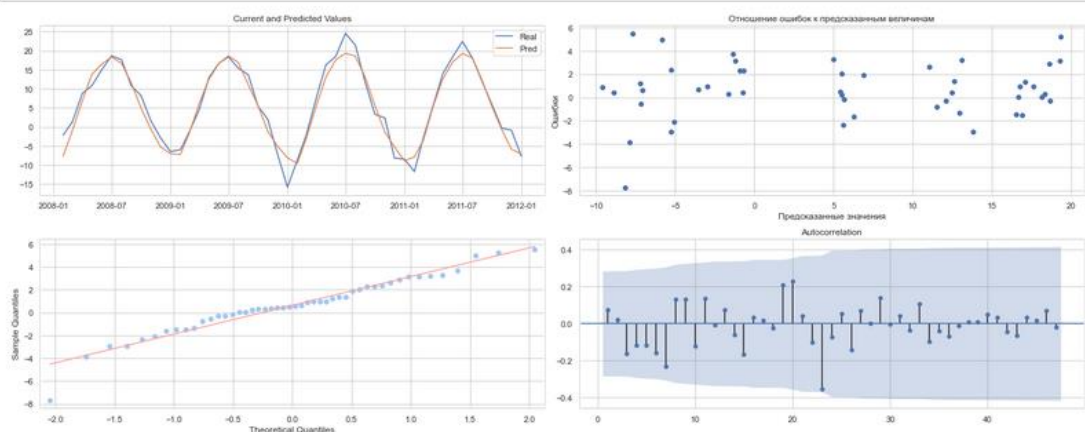
plot_acf(data.iloc[:,2], lags=(len(data.iloc[:,2])-1), zero=False, ax=ax4)
plt.tight_layout()
plt.show()
```

```
In [32]: # Удаляем некоторые столбцы для построения графиков
val.drop(['month', 'year'], axis=1, inplace=True)
val.head()
```

```
Out[32]:
```

	Temp	Pred	Error
Date			
2008-01-31	-2.197	-7.692289	5.495289
2008-02-29	1.340	-0.934842	2.274842
2008-03-31	8.806	6.896832	1.909168
2008-04-30	10.868	13.796154	-2.928154
2008-05-31	15.058	16.519870	-1.461870

```
In [33]: plot_error(val)
```



Анализируя вышеприведенные графики, мы видим, что прогнозы очень хорошо соответствуют текущим значениям.

Погрешность в сравнении с прогнозируемыми значениями имеет линейное распределение (погрешности составляют от -8 до +6 при повышении температуры).

Левый нижний график показывает нормальный паттерн с небольшими выбросами.

График автокорреляции показывает положительный всплеск доверительного интервала чуть выше второго лага, но я считаю, что больше не нужно никаких изменений.

Наконец пришло время экстраполировать прогноз в наборе тестов за последние 12 месяцев:

```
In [34]: # Создание новой конкатенации обучающего и проверочного набора:
future = pd.concat([train['Temp'], val['Temp']])
future.head()

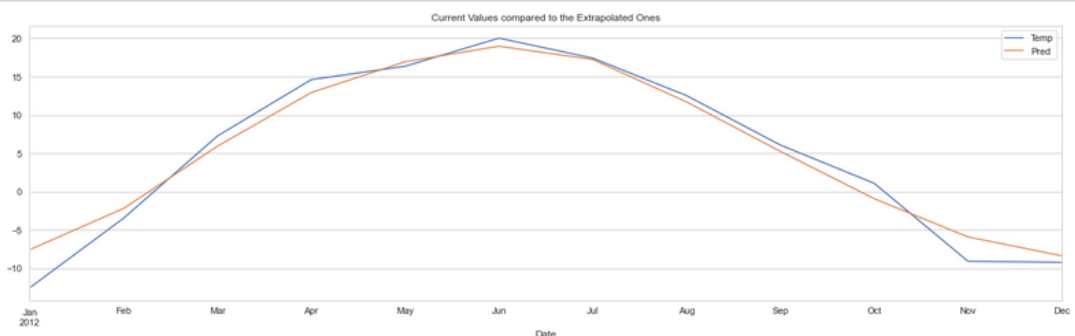
Out[34]: Date
1900-01-31    -10.109
1900-02-28     -4.419
1900-03-31     2.300
1900-04-30     9.853
1900-05-31    13.814
Name: Temp, dtype: float64

In [35]: # Используем те же параметры подогнанной модели
model = sm.tsa.statespace.SARIMAX(future, order=(3,0,0), seasonal_order=(0,1,1,12), trend='c')
result = model.fit(dispatch=False)
```

Создаем новый столбец в тестовом наборе с прогнозируемыми значениями и сравниваем их с реальными значениями.

```
In [36]: test['Pred'] = result.predict(start=(len(future)), end=(len(future)+13))

In [37]: test[['Temp', 'Pred']].plot(figsize=(22,6))
plt.title('Current Values compared to the Extrapolated Ones')
plt.show()
```



Похоже, что параметры SARIMA были хорошо подогнаны, прогнозируемые значения соответствуют реальным значениям, а также сезонной структуре.

Заключение

Данная работа показывает, как важен опыт прошлого. Ведь с современными технологиями мы можем предотвратить последствия климатических изменений. Если применить данную модель с более точными данными распределения температуры для других городов России, мы можем получить более точный план примерного изменения температуры. Точное прогнозирование изменения климата поможет людям предвидеть такие катаклизмы, как массовые пожары лесов, наводнение прибрежных к рекам городов и т.д.

Наконец, я оценив модель с RMSE в тестовом наборе (baseline против экстраполяции):

```
In [40]: test_baseline = test['Temp'].shift()

test_baseline[0] = test['Temp'][0]

rmse_test_base = measure_rmse(test['Temp'], test_baseline)
rmse_test_extrap = measure_rmse(test['Temp'], test['Pred'])

print(f'RMSE baseline для теста baseline составило {round(rmse_test_base,2)} градуса Цельсия')
print(f'RMSE baseline для теста экстраполяции составило {round(rmse_test_extrap,2)} градуса Цельсия')
print(f'Улучшение составляет {-round((rmse_test_extrap/rmse_test_base-1)*100,2)}%')
```

RMSE baseline для теста baseline составило 6.24 градуса Цельсия
RMSE baseline для теста экстраполяции составило 2.01 градуса Цельсия
Улучшение составляет 67.81%

Можно сказать, насколько эффективна данная модель.

Список литературы

1. Информационные ресурсы по теме SARIMA:
<https://machinelearningmastery.com/sarima-for-time-series-forecasting-in-python/>
https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average
2. https://en.wikipedia.org/wiki/Root-mean-square_deviation
3. Из курса лекций по основе Анализа данных «Модель авторегрессии и скользящего среднего ARMA(p,q)» Борис Демешев (ВШЭ, Москва):
<https://www.youtube.com/watch?v=pQm7ZDgB1tA>
4. «Лекционные материалы по курсу ТМО» Ю.Е. Гапанюк:
https://github.com/ugapanyuk/ml_course_2020/wiki/COURSE_TMO
5. «Методические указания к курсовой работе» Ю.Е. Гапанюк:
https://nbviewer.jupyter.org/github/ugapanyuk/ml_course_2020/blob/master/common/notebooks/projectexample/project_clas_regr.ipynb