NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS

Faculty of Computer Science
Bachelor’s Programme “Data Science and Business Analytics”

Software Project Report on the Topic:
Assessing greenhouse emission with machine learning

Fulfilled by:
Student of the Group БПАД213
Dineev Ilshat Rishatovich

	[illegible]	09.06.2023
	(signature)	*(date)*

Checked by the Project Supervisor:

Petrov Petr Petrovich
Senior Lecturer
Faculty of Computer Science, HSE University

Rodriges		*09.06.2023*
(signature)		*(date)*

Moscow 2023

Contents

Annotation

Greenhouse emission is one of the biggest problems of the modern world and it’s very crucial to analyze them to decrease their impact on our future life .This project is aimed to assess greenhouse emission using different methods of machine learning. This approach to this problem provides better understanding of different patterns of emission and analyzing big volumes of data

Аннотация

Выбросы парниковых газов одна из самых больших проблем современного и очень важно анализировать их чтобы уменьшить их влияние на нашу жизнь в будущем. Эта работа направлена на оценка выбросов парниковых газов используя разные методы машинного обучения. Такой подход к проблеме предоставляет лучшее понимание разных шаблонов выбросов и позволяет анализировать большие объемы информации.

Key words

Greenhouse emissions, machine learning (ML), machine learning model, neural networks, satellite data.

Introduction

Description of subject field

Greenhouse emission usually refers to release of greenhouse gasses to the atmosphere. Gasses such as CO2 and CH4 play the greatest role in this process. Their emissions affect global warming and climate change. To get information about emissions, data from satellites are usually used, which collect information about the earth's surface, including the above gasses. The amount of information from satellites is quite large, so machine learning algorithms are usually used to process the data received from them.

Machine learning is a field that is used to build models to find patterns and relations on some datasets. Machine learning uses different statistical methods to perform this tasks

Relevance

One of the biggest problems of humanity today is global warming and climate change. So it is very important to investigate this issue from all sides and try to find any patterns in this process, which machine learning algorithms can help with

Additionally, machine learning can be used to develop predictive models that can be used to forecast future emissions and inform mitigation strategies.

Also, such a way of considering greenhouse gas emission problem helps us to exclude various speculations on the topic of the greenhouse effect, since it relies on statistical methods and leaves no room for a subjective view

This project has a big value for me, as it will allow me to learn machine learning and improve my skills in this field

Goal

The goal of the project is to learn machine learning, develop model of greenhouse emissions, identify patterns in the data, make a correct assessment based on the result obtained

Tasks

- Learn Machine learning
- Find dataset about greenhouse emission

- Develop machine learning model
- Check this model on the test sample
- Create Jupyter notebook with illustration of the obtained results
- Assess results of prediction made by ML model

The importance of CO2 mapping

CO2 mapping is very important because it allows you to analyze gas emissions at the global, regional and local levels. This allows you to very accurately determine the regions that produce the most CO2. Also, with the help of mapping, you can clearly see the distribution of CO2 on Earth. Thus, it is possible to identify regions with low, medium and high emissions of this greenhouse gas.

Description of functional and non-functional requirements

Functional requirements

1. Data collection and preprocessing: Gathering and preparing data about greenhouse emission
2. Model selection: Choosing the appropriate model
3. Model training: Training the model on the data
4. Model evaluation: Evaluating the performance of the model
5. Performing results: Building correct visualization of the obtained results

Non-functional requirements

1. Model must be built in satisfactory time
2. Model should be able to handle increasing amounts of data.

1. The current state of the problem and the formulation of the problem

1.1. Forecast of CO2 concentration

Usually when we talk about the concentration of CO2, we mean the concentration of CO2 in the lower layer of the atmosphere, called the troposphere, located at a distance of 8 - 15 km from the earth's surface. The concentration of CO2 in the troposphere is measured in particles per million (ppm), particles per billion (ppb), and so on. The mole fraction of dioxide measurement is also used.

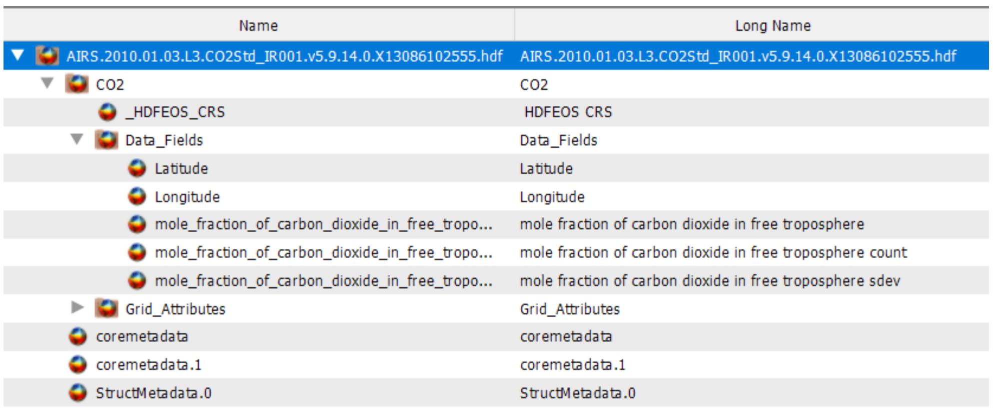
Different approaches to measuring CO2 are used. One of the methods is measurements using a ground-based meteorological station, which provides point information about the amount of CO2 at a specific point. The data received from the satellite is also used. This method allows you to find out the concentration of CO2 at each point of the earth

Measuring and forecasting the concentration of CO2 in the troposphere is very important for understanding the processes of climate change and global warming. Subsequently, it will be possible to take any steps to combat this problem based on these data.

1.2. Remote sensing data of the Earth

Remote sensing is the acquisition of information from a distance. NASA observes Earth and other planetary bodies via remote sensors on satellites and aircraft that detect and record reflected or emitted energy. Remote sensors, which provide a global perspective and a wealth of data about Earth systems, enable data-informed decision making based on the current and future state of our planet.[1]

Remote sensing data, as the most of scientific data, is recorded in the HDF (hierarchical data format). As we can see from its name, the main feature of this format is hierarchical form.



Name	Long Name
AIRS.2010.01.03.L3.CO2Std_IR001.v5.9.14.0.X13086102555.hdf	AIRS.2010.01.03.L3.CO2Std_IR001.v5.9.14.0.X13086102555.hdf
CO2	CO2
_HDFEOS_CRS	HDFEOS CRS
Data_Fields	Data_Fields
Latitude	Latitude
Longitude	Longitude
mole_fraction_of_carbon_dioxide_in_free_tropo...	mole fraction of carbon dioxide in free troposphere
mole_fraction_of_carbon_dioxide_in_free_tropo...	mole fraction of carbon dioxide in free troposphere count
mole_fraction_of_carbon_dioxide_in_free_tropo...	mole fraction of carbon dioxide in free troposphere sdev
Grid_Attributes	Grid_Attributes
coremetadata	coremetadata
coremetadata.1	coremetadata.1
StructMetadata.0	StructMetadata.0

Figure 1.1 - Structure of HDF file

From this picture, we can see that file have variables - in our example it's Latitude, Longitude, mole fraction of carbon dioxide in free troposphere, mole fraction of carbon dioxide in free troposphere count, mole fraction of carbon dioxide in free troposphere sdev.

1.3 Modern approaches to CO2 concentration prediction

There are several approaches to predict CO2 concentration.

1. Machine learning. ML methods can be applied for processing and finding complex relationships between data. ML works on statistical methods, such as gradient boosting [2], Ordinary least squares method [3], linear and non linear regressions [4][5]. Also ML models support work with time series, which will allow us to find seasonal trends and cycles. Example of such approach can be found in work Forecasting CO2 Emission with Machine Learning Methods[6]
2. Neural networks. Artificial NNs are also a very popular instrument to predict greenhouse emission. Neural networks process large amounts of historical data through interconnected artificial neurons and activation functions. For solving problem of this type, can be applied different types of NN: Convolutional neural networks, recurrent neural networks. Such approach was implemented in the work A Neural Network Model for Forecasting CO2 Emission [7]
3. Deep learning. This technology is using deep neural networks with multiple layers. This allows to find complex relationships. Implementation of this approach depends on the type of the data. So, for spatial data convolutional networks are most suitable, whereas for time series recurrent NN is the better. Deep learning method was implemented in the work A deep learning-based direct forecasting of CO2 plume migration [8]

1.4 Conclusions on the section and problem statement

After studying the methods of predicting greenhouse gas emissions, you can identify the problem of the project and determine the steps to solve it. So, to choose a method for predicting greenhouse gasses, you need to select and determine the shape of the dataset, how it was assembled and what it shows.

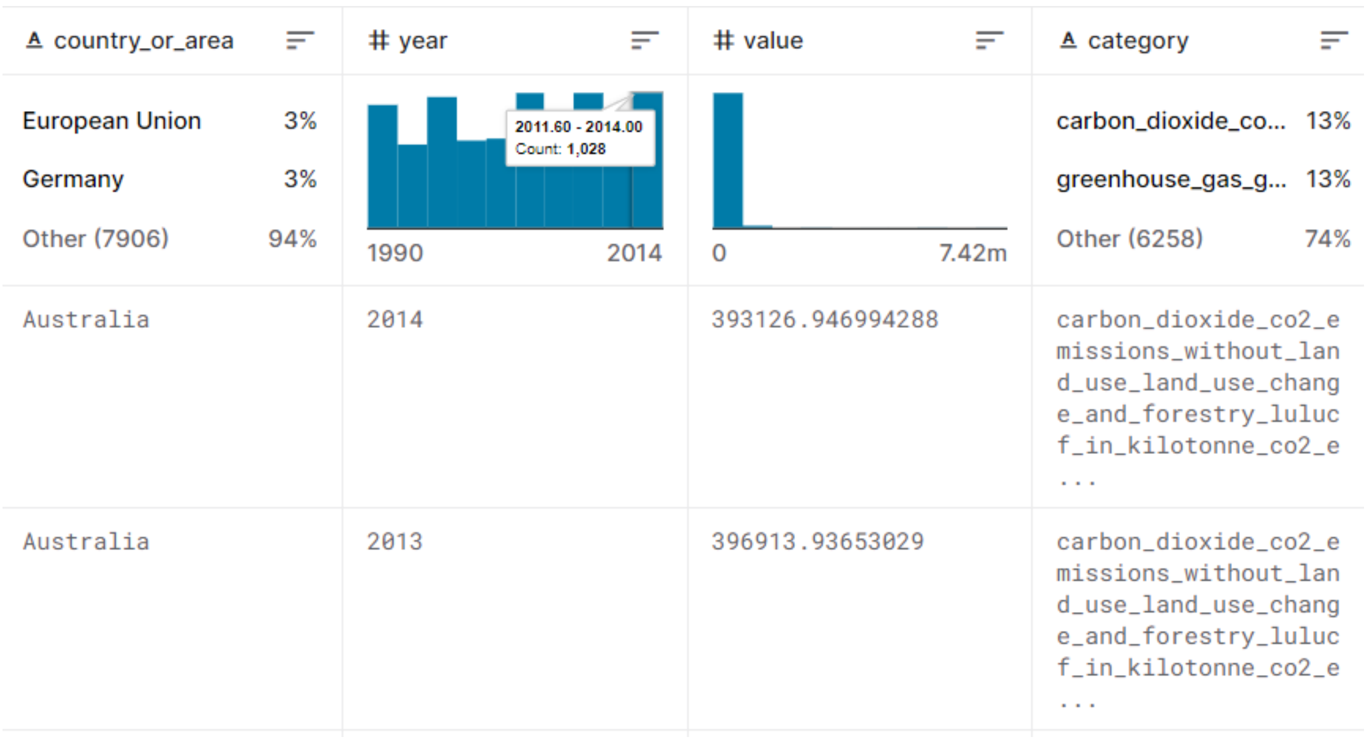
Plan of project:

1. Find dataset about greenhouse emission
2. Study its form and the data it represents
3. Create model
4. Formulate quality metrics
5. Evaluate results

2. Prediction of the greenhouse emission using machine learning

2.1. Exploration of dataset

I have chosen International Greenhouse Gas Emissions from Kaggle.com. Information in this dataset was collected from open sources. Big advantage of it is that it gives information by every country and by every type of greenhouse emission.



country_or_area	year	value	category
European Union 3% Germany 3% Other (7906) 94%	2011.60 - 2014.00 Count: 1,028 1990 2014	0 7.42m	carbon_dioxide_co... 13% greenhouse_gas_g... 13% Other (6258) 74%
Australia	2014	393126.946994288	carbon_dioxide_co2_emissions_without_land_use_land_use_change_and_forestry_lulucf_in_kilotonne_co2_e...
Australia	2013	396913.93653029	carbon_dioxide_co2_emissions_without_land_use_land_use_change_and_forestry_lulucf_in_kilotonne_co2_e...

Figure 2.1. - International Greenhouse emissions dataset

Let’s take a look at the one country more closely

```
country_or_area  year         value  \           category
0        Australia  2014  393126.946994   carbon_dioxide_co2_emissions_without_land_use_...
1        Australia  2013  396913.936530   carbon_dioxide_co2_emissions_without_land_use_...
2        Australia  2012  406462.847704   carbon_dioxide_co2_emissions_without_land_use_...
3        Australia  2011  403705.528314   carbon_dioxide_co2_emissions_without_land_use_...
4        Australia  2010  406200.993184   carbon_dioxide_co2_emissions_without_land_use_...
...              ...   ...            ...
7320     Australia  1993     264.759156   sulphur_hexafluoride_sf6_emissions_in_kilotonn...
7321     Australia  1992     246.858849   sulphur_hexafluoride_sf6_emissions_in_kilotonn...
7322     Australia  1991     228.944834   sulphur_hexafluoride_sf6_emissions_in_kilotonn...
7323     Australia  1990     211.018511   sulphur_hexafluoride_sf6_emissions_in_kilotonn...
```

Here we can find one interesting feature of this data set: data about the country is given not only by every year, but also by every category of emission. That means we will need to build many models, for every category of emission, to give more accurate predictions. As if we will unite categories and sum value by every year, we won't take into account that different categories may have different behavior.

Also let's take a look at the relation between year and value of emission.

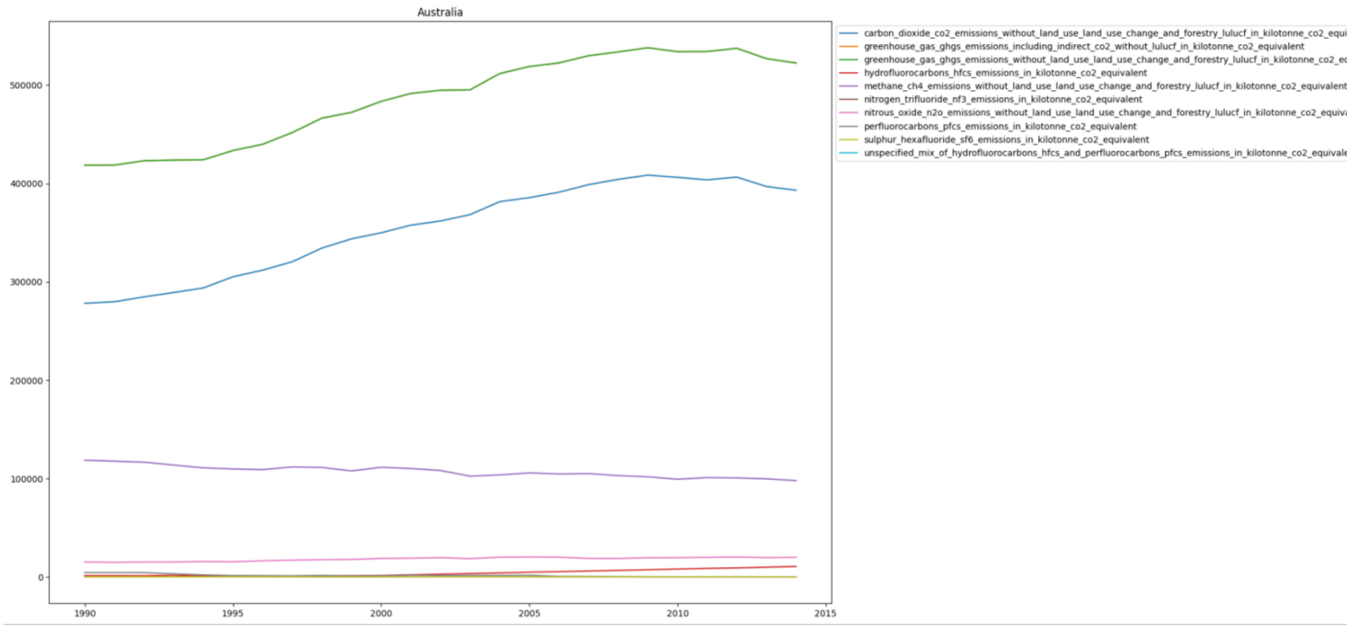


Figure 2.2 - Relation between emission and year by every category in Australia

As we can see from this graph, the relation between emission and year is not linear, so we need to use a machine learning model based on the nonlinear regression. So we have two most appropriate ways: using a Random Forest regressor or model based on the Time Series. Let's first try Random Forest

2.2. Random Forest Regressor

Random Forest[18] is technology which fits a number of decision trees on sub-samples of the dataset and by averaging models provides best predictive results. As I said before, Random Forest regressor can work with non-linear relations.

To solve this problem, I made the next preprocessing: grouped dataset by country and category of emission:

```
# Group the data by country and category
grouped_data = data.groupby(['country_or_area', 'category'])
```

Then for every group I define variables and target values. In my case, X part is year, Y - value of emission. I divide X and Y into training and test parts, and then train models on this data. For this metrics I chose MSE as metrics. After that’s all, I made predictions for future years. Here’s the code:

```
    # Loop through each group
    for (country, category), group in grouped_data:
        # Separate the features (year) and target variable (value)
        X = group[['year']]
        y = group['value']

        # Split the data into training and test sets
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

        # Create and fit the random forest model
        model = RandomForestRegressor(n_estimators = 50, random_state=42)
        model.fit(X_train, y_train)

        # Predict the values for the next years
        future_years = pd.DataFrame({'year': [2015, 2016, 2017]})  # Adjust the
years as needed
        category_predictions = model.predict(future_years)

        # Calculate MSE between predicted values and actual test values
        mse = mean_squared_error(y_test, model.predict(X_test))
        rmse = mse ** 0.5

        # Store the predictions, MSE, country, category, and year
        for year, prediction in zip(future_years['year'], category_predictions):
            predictions.append({
                'country': country,
                'category': category,
                'year': year,
                'value': prediction,
                'mse': mse,
                'rmse': rmse
            })

    # Create a DataFrame for the predictions
    predictions_df = pd.DataFrame(predictions)

    # Print the predicted values and MSE for each country, category, and year
    print(predictions_df)
```

In the result, I get next output:

```
                    country    category                          year    value
0                  Australia   carbon_dioxide_co2_emissions…    2015  399215.503001
1                  Australia   carbon_dioxide_co2_emission…     2016  399215.503001
2                  Australia   carbon_dioxide_co2_emission…     2017  399215.503001
3                  Australia   greenhouse_gas_ghgs_emissi…      2015  529206.46787
4                  Australia   greenhouse_gas_ghgs_emissi…      2016  529206.467879
...                     ...
```

As we can see, the model gives the same value for different years. This mistake is connected to the features of Random Forest regressor. It can give appropriate predictions only for X, that is located in the training range. In our case, we try to predict future years, which are out of range of the training part. In the result I can say that I need to try a different approach.

2.3. Time series forecasting

My dataset is actually a time series[16], so this approach will be very suitable. To solve this problem I will use Facebook library Prophet [9]. For preprocessing I will use the same approach as in the previous model, but I need to change year to datetime format:

```
    group['ds'] = pd.to_datetime(group['ds'], format='%Y')    # convert year to 
datetime
```

Also as a result of experiments, I decided not to split the model into training and test parts, as the last 4 years have begun a trend of decreasing emission, so it’s very crucial to take it into account.

There’s code of the model:

```
    import pandas as pd
    from prophet import Prophet
    import logging, sys
    logging.disable(sys.maxsize)
    # Load the dataset
    data                                                                           = 
pd.read_csv(r'C:\Users\idine\OneDrive\Документы\Курсовая\greenhouse_gas_inventory_
data_data(1).csv')

    # List to store the predicted values and MSE for each country
    predictions = []

    # Group the data by country and category
    grouped_data = data.groupby(['country_or_area', 'category'])

    # Loop through each group
    for (country, category), group in grouped_data:
        # Separate the features (year) and target variable (value)
        group = group.rename(columns={'year': 'ds', 'value': 'y'})
        group['ds'] = pd.to_datetime(group['ds'], format='%Y')   # convert year to 
datetime

        # Create and fit the Prophet model
        model = Prophet()
        model.fit(group)

        # Predict the values for the next years
        future_years = model.make_future_dataframe(periods=15, freq='Y')   # Adjust 
the years as needed
        forecast = model.predict(future_years)

        # Store the predictions, country, category, and year
        for row in forecast[['ds', 'yhat']].itertuples():
            if row.ds.year in [2015, 2016, 2017, 2018, 2019]:
                predictions.append({
                    'country': country,
                    'category': category,
                    'year': row.ds.year,
                    'value': row.yhat,
                })

    # Create a DataFrame for the predictions
    predictions_df = pd.DataFrame(predictions)
```

Let’s visualate and evaluate results for one of the countries:

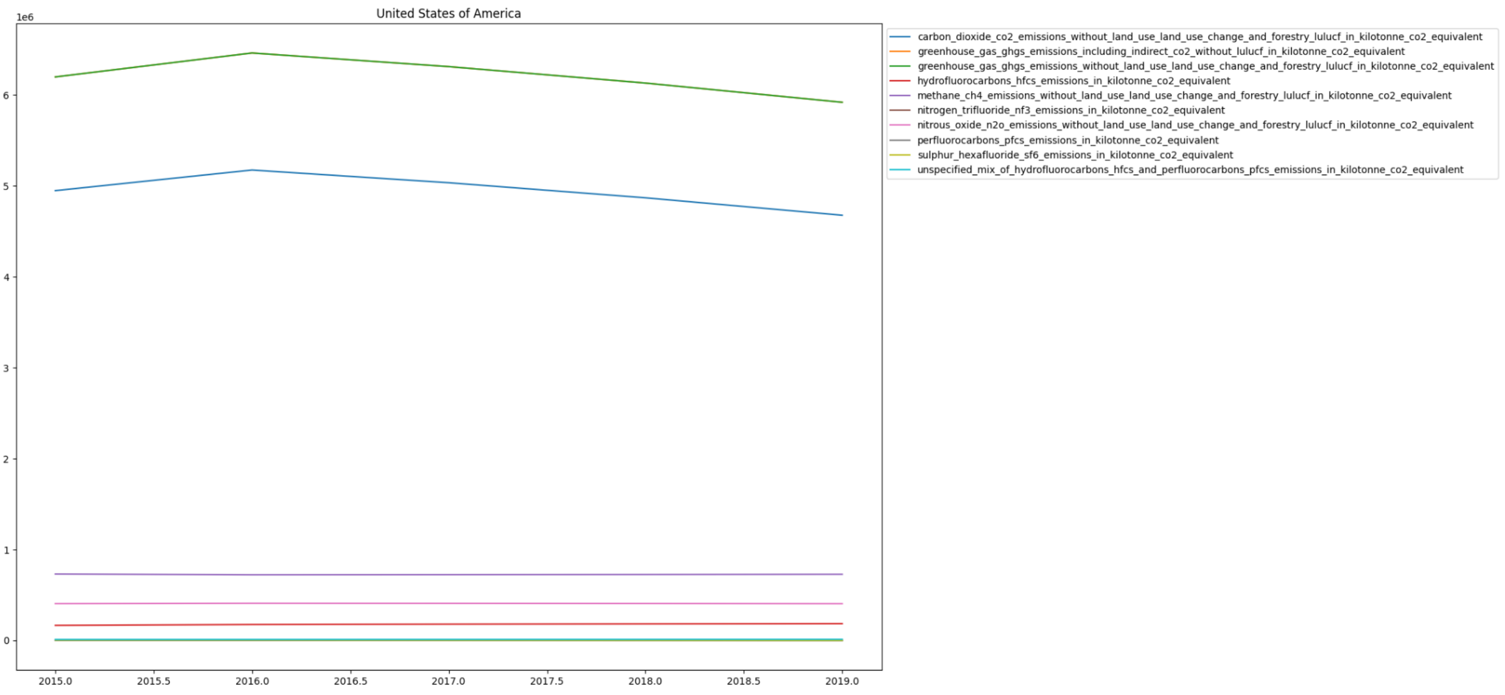


Figure 2.3 - Greenhouse Emission prediction for the United States.

As we can see from this graph, in the United States of America, emissions will decrease significantly in the next few years. We can assume that this is connected to the government policy of ecology protection. Anyway, we can't deny that the trend of decreasing is there.

3. Spatial data and technologies

3.1. Description of dataset

To solve this problem, my scientific supervisor suggested that I take a dataset about CO2 emission from a NASA satellite [10]. This dataset is represented as 1646 HDF 4 files, containing information about CO2 emission. It has 5 variables: Latitude, Longitude, mole fraction of carbon dioxide in free troposphere, mole fraction of carbon dioxide in free troposphere count, mole fraction of carbon dioxide in free troposphere sdev. From all these variables we need only 3 first, as they provide information I need. Here's visualization of CO2 variable:

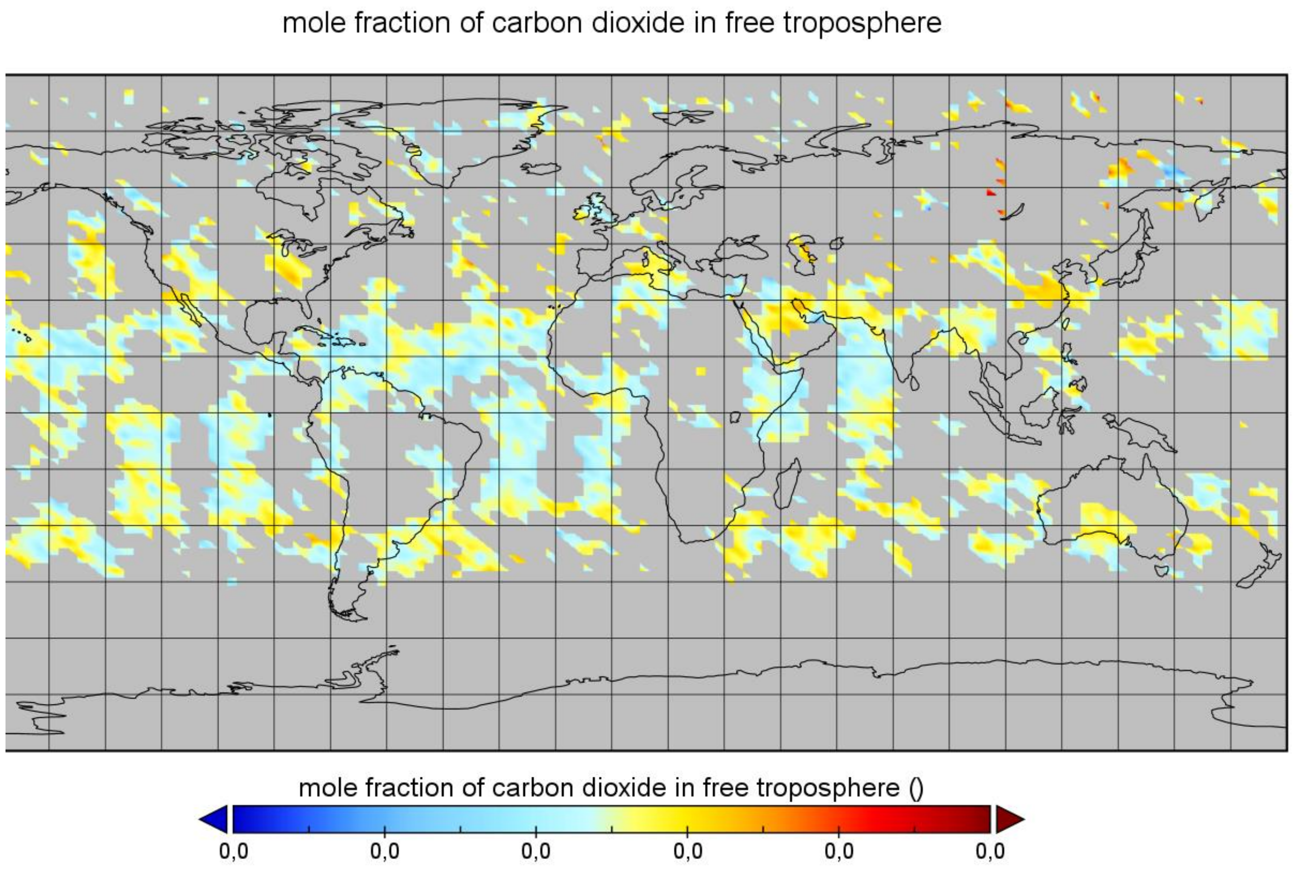
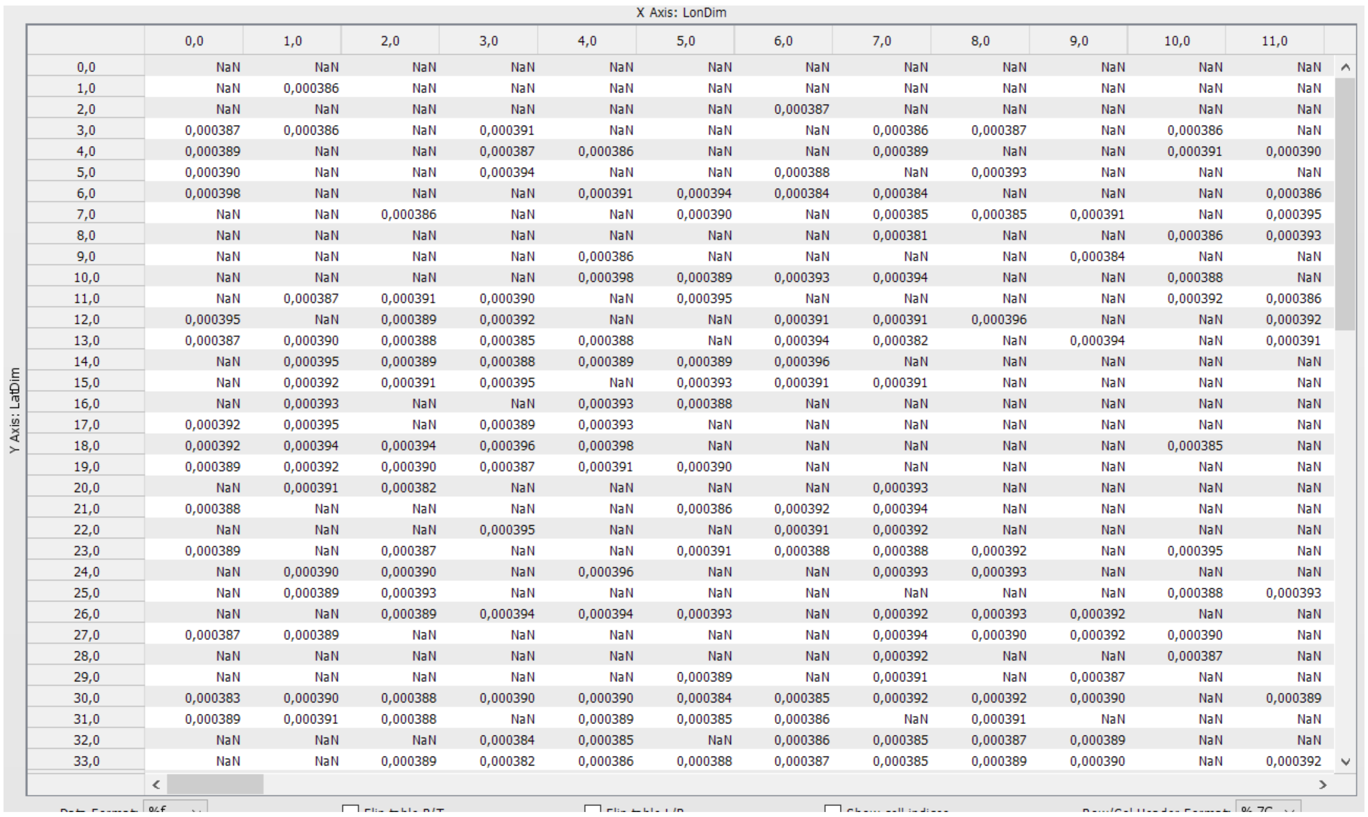

mole fraction of carbon dioxide in free troposphere
mole fraction of carbon dioxide in free troposphere ()
0,0
0,0
0,0
0,0
0,0
0,0

Figure 3.1 - CO2 emission visualization

As we can see, information in this variable is CO2 emission by coordinates. With other 2 variables we can define coordinates that correspond to the CO2 emission. All 3 variables are 2-dimensional arrays, by comparing values with the same indices, it is possible to determine the value of CO2 emissions for given latitude and longitude values. Whole dataset has a shape (1646, 3, 91, 144). First dimension is day, second is variable, third and fourth are values of variables. Here's one of the tables with CO2 emission:



X Axis: LonDim

Y Axis: LatDim

	0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0	10,0	11,0
0,0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1,0	NaN	0,000386	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2,0	NaN	NaN	NaN	NaN	NaN	NaN	0,000387	NaN	NaN	NaN	NaN	NaN
3,0	0,000387	0,000386	NaN	0,000391	NaN	NaN	NaN	0,000386	0,000387	NaN	0,000386	NaN
4,0	0,000389	NaN	NaN	0,000387	0,000386	NaN	NaN	0,000389	NaN	NaN	0,000391	0,000390
5,0	0,000390	NaN	NaN	0,000394	NaN	NaN	0,000388	NaN	0,000393	NaN	NaN	NaN
6,0	0,000398	NaN	NaN	NaN	0,000391	0,000394	0,000384	0,000384	NaN	NaN	NaN	0,000386
7,0	NaN	NaN	0,000386	NaN	NaN	0,000390	NaN	0,000385	0,000385	0,000391	NaN	0,000395
8,0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0,000381	NaN	NaN	0,000386	0,000393
9,0	NaN	NaN	NaN	NaN	0,000386	NaN	NaN	NaN	NaN	0,000384	NaN	NaN
10,0	NaN	NaN	NaN	NaN	0,000398	0,000389	0,000393	0,000394	NaN	NaN	0,000388	NaN
11,0	NaN	0,000387	0,000391	0,000390	NaN	0,000395	NaN	NaN	NaN	NaN	0,000392	0,000386
12,0	0,000395	NaN	0,000389	0,000392	NaN	NaN	0,000391	0,000391	0,000396	NaN	NaN	0,000392
13,0	0,000387	0,000390	0,000388	0,000385	0,000388	NaN	0,000394	0,000382	NaN	0,000394	NaN	0,000391
14,0	NaN	0,000395	0,000389	0,000388	0,000389	0,000389	0,000396	NaN	NaN	NaN	NaN	NaN
15,0	NaN	0,000392	0,000391	0,000395	NaN	0,000393	0,000391	0,000391	NaN	NaN	NaN	NaN
16,0	NaN	0,000393	NaN	NaN	0,000393	0,000388	NaN	NaN	NaN	NaN	NaN	NaN
17,0	0,000392	0,000395	NaN	0,000389	0,000393	NaN	NaN	NaN	NaN	NaN	NaN	NaN
18,0	0,000392	0,000394	0,000394	0,000396	0,000398	NaN	NaN	NaN	NaN	NaN	0,000385	NaN
19,0	0,000389	0,000392	0,000390	0,000387	0,000391	0,000390	NaN	NaN	NaN	NaN	NaN	NaN
20,0	NaN	0,000391	0,000382	NaN	NaN	NaN	NaN	0,000393	NaN	NaN	NaN	NaN
21,0	0,000388	NaN	NaN	NaN	NaN	0,000386	0,000392	0,000394	NaN	NaN	NaN	NaN
22,0	NaN	NaN	NaN	0,000395	NaN	NaN	0,000391	0,000392	NaN	NaN	NaN	NaN
23,0	0,000389	NaN	0,000387	NaN	NaN	0,000391	0,000388	0,000388	0,000392	NaN	0,000395	NaN
24,0	NaN	0,000390	0,000390	NaN	0,000396	NaN	NaN	0,000393	0,000393	NaN	NaN	NaN
25,0	NaN	0,000389	0,000393	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0,000388	0,000393
26,0	NaN	NaN	0,000389	0,000394	0,000394	0,000393	NaN	0,000392	0,000393	0,000392	NaN	NaN
27,0	0,000387	0,000389	NaN	NaN	NaN	NaN	NaN	0,000394	0,000390	0,000392	0,000390	NaN
28,0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0,000392	NaN	NaN	0,000387	NaN
29,0	NaN	NaN	NaN	NaN	NaN	0,000389	NaN	0,000391	NaN	0,000387	NaN	NaN
30,0	0,000383	0,000390	0,000388	0,000390	0,000390	0,000384	0,000385	0,000392	0,000392	0,000390	NaN	0,000389
31,0	0,000389	0,000391	0,000388	NaN	0,000389	0,000385	0,000386	NaN	0,000391	NaN	NaN	NaN
32,0	NaN	NaN	NaN	0,000384	0,000385	NaN	0,000386	0,000385	0,000387	0,000389	NaN	NaN
33,0	NaN	NaN	0,000389	0,000382	0,000386	0,000388	0,000387	0,000385	0,000389	0,000390	NaN	0,000392

Figure 3.2 - Table of CO2 emissions values

3.2 Downloading data.

To work this dataset I saved it to a numpy array, as it provides best speed. To download this data, I added all files to the folder, wrote a regular expression to extract the date from the filename. Then added every variable into the corresponding dimension. Also I deleted values about the south part of Earth, because satellites can't see this part and there are only empty values. Here's the code of this algorithm:

# Location of your hdf files

folder = r"C:\Users\idine"

```
# Variable names in the hdf files
variables = [
    "Latitude",
    "Longitude",
    "mole_fraction_of_carbon_dioxide_in_free_troposphere"
]

# Extract date from filename
def get_date_from_filename(filename):
    date_str = re.findall(r'\d{4}\.\d{2}\.\d{2}', filename)[0]
    return datetime.datetime.strptime(date_str, "%Y.%m.%d")

# Get list of all hdf files in the folder
files = [f for f in os.listdir(folder) if f.endswith(".hdf")]
```

```
    # Sort files by date
    files.sort(key=get_date_from_filename)

    # Initialize list to hold data and dates
    data_list = []
    date_list = []

    # Loop through all sorted hdf files in the folder
    for filename in files:
        file = os.path.join(folder, filename)

        # Open the hdf file
        hdf = SD(file, SDC.READ)

        # Temporarily store the variables
        temp_data = []

        # Loop through each variable and append the data to the corresponding list
        for var in variables:
            temp_data.append(hdf.select(var)[:])

        # Append temporary data to main list
        data_list.append(temp_data)

        # Get the date from the filename and convert to timestamp
        date = get_date_from_filename(filename)
        timestamp = date.timestamp()

        # Add timestamp to the date list
        date_list.append(timestamp)

        # Close the file
        hdf.end()

    # Convert lists to numpy arrays
    data_np = np.array(data_list)
    date_np = np.array(date_list)

    # Create a dictionary to store both arrays
    data_dict = {"dates": date_np, "data": data_np}
```

3.3. Hypothesis on CO2 emission

There are several hypothesis that I want to check in this project:

1. CO2 emission bigger in the industrial zones
2. Cities with a population of more than a million people produce as much as in the big industrial zones
3. In the forests CO2 emission is small
4. The biggest emission is around heavy production

4. Forecasting CO2 emission.

4.1 The peculiarity of the proposed approach

For work with this dataset I decided to use neural networks, as they provide analyzing spatial data. First of all, as the baseline I chose a multilayer perceptron, I wanted to predict emission of the 8th day by 7 previous days. To do this I flattened a matrix of CO2 values, splitted dataset to a combination of [i+8] and [i:i+7] days. In the result, as perceptrons don't take into account spatial aspects of the data, so it predicted values by places in the vector, and didn't take into account nearest values, that's why prediction became "striped", which was incorrect.

To solve this problem I added Convolutional layers. Feature of this approach is that it has a 2 dimensional kernel[20] that goes through the matrix and takes into account the spatial aspect of the data, so this method is very suitable. After that, our matrix becomes smaller, and in the result, we can vector of values from our matrix that represents the spatial aspect of the data, and then we can use a linear layer to this vector.

4.2. The general scheme of the approach

At the input, the model should get an expanded data set. After that, I need to create X and Y, where Y is the next day, and X is the current day, so we will predict future days by previous ones. Then we will split X and Y in batches. After that we give these batches to model. Here's illustration of the model algorithm:

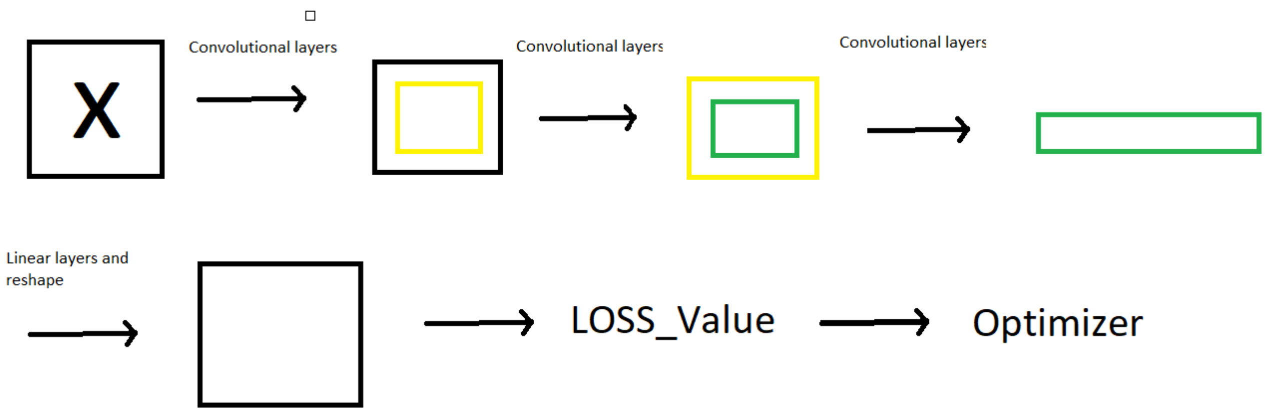


Figure 4.1 - Algorithm of model

After this algorithm, we will get a model that will give a prediction matrix for the next date.

4. 3. Data preprocessing

Satellites are very sensitive to clouds, that's why there are some missing values in the data. To solve this problem there are several, such as approximation, interpolation[17] and leaving missing data. After learning the theme of working with missing values, I decided to use an approach called temporal spatial interpolation[15]. Main feature of this technology is that it takes into account both time and spatial aspects of the data, so it's sensitive to previous and nearest values. So that's why this approach is the best approach. Here's the code:

```
    # Temporally interpolate for each latitude/longitude
    for i in range(data_np.shape[2]):
        for j in range(data_np.shape[3]):
            valid_mask = np.isfinite(data_np[:, target_var_index, i, j])
            if np.any(valid_mask):
                interp_fn     =     interpolate.interp1d(np.flatnonzero(valid_mask),
data_np[valid_mask, target_var_index, i, j], fill_value="extrapolate")
                data_np[:,        target_var_index,        i,        j]        =
interp_fn(np.arange(data_np.shape[0]))

    # Now, spatially interpolate for each day
    for t in range(data_np.shape[0]):
        valid_mask = np.isfinite(data_np[t, target_var_index])
        if np.any(valid_mask):
            coords_known = np.column_stack(np.nonzero(valid_mask))
            values_known = data_np[t, target_var_index][valid_mask]
            coords_interp = np.column_stack(np.nonzero(~valid_mask))
            data_np[t,              target_var_index][~valid_mask]              =
interpolate.griddata(coords_known, values_known, coords_interp, method='nearest')
```

Then I scaled[19] data to improve performance of the model.

```
    # Reshape for scaler
    target = target.reshape(-1, 1)

    # Normalize the target variable
    scaler = MinMaxScaler(feature_range=(0, 1))
    target_scaled = scaler.fit_transform(target)
    # Return scaled data to original form

    print(target_scaled)transformed_data = np.copy(data_np)
    target_scaled_reshaped        =        target_scaled.reshape(data_np.shape[0],
data_np.shape[2], data_np.shape[3])
    transformed_data[:, target_var_index, :, :] = target_scaled_reshaped
    print(transformed_data[0][0])
```

I decided to take not all dataset, but some definite region. Here's this area on the map:

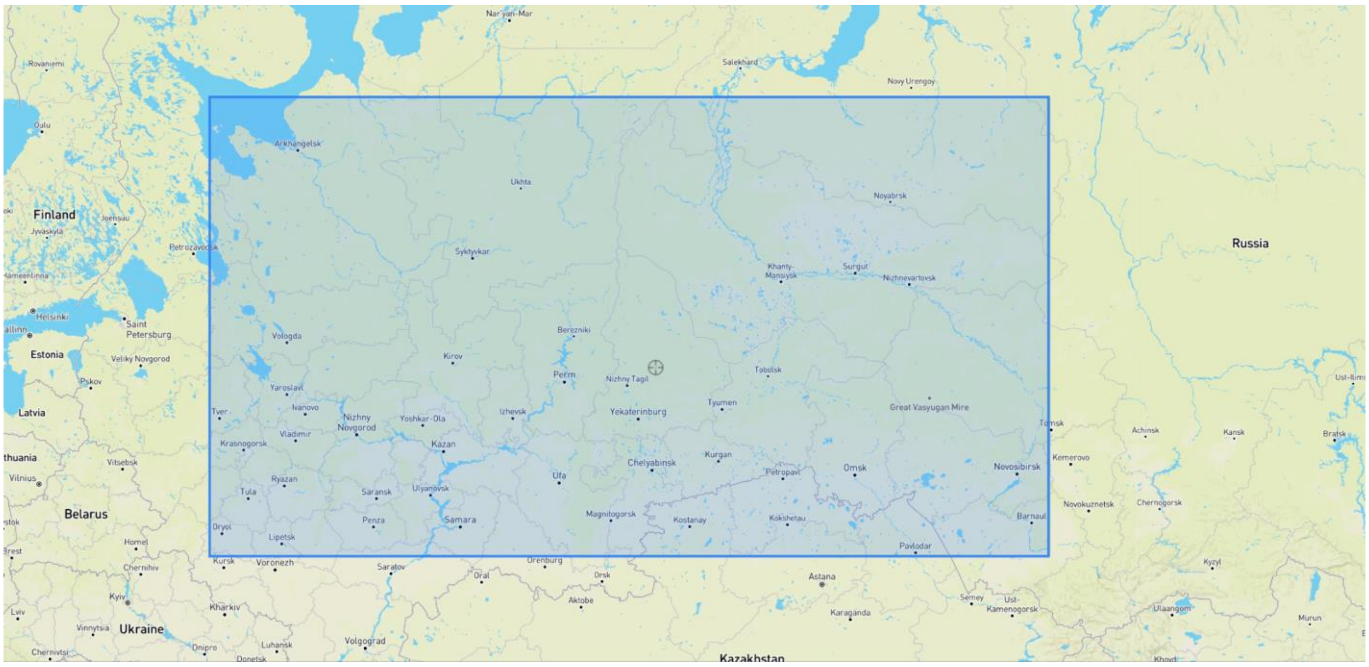


Figure 4.2 - Working area

To take this area from the dataset, I created a mask that is bounded by minimum and maximum values of longitude and latitude. Then I made slice by this mask:

```
    # Define your longitude and latitude boundaries
    lon_min, lon_max = 35, 85
    lat_min, lat_max = 52, 65
    from scipy import stats as st
    # Combine latitudes and longitudes to 2D coordinates
    coordinates = np.column_stack((latitudes.flatten(), longitudes.flatten()))

    # Define the region boundaries
    region = np.array([[lat_min, lon_min], [lat_max, lon_max]])

    # Get the coordinates that fall within the region
    region_mask = (coordinates[:, 0] >= region[0, 0]) & (coordinates[:, 0] <=
region[1, 0]) & \
                  (coordinates[:, 1] >= region[0, 1]) & (coordinates[:, 1] <=
region[1, 1])

    # Get the indices of these coordinates
    region_indices = np.nonzero(region_mask)

    # Reshape indices to the shape of latitude/longitude arrays
    region_indices_reshaped = np.unravel_index(region_indices, latitudes.shape)

    # Slice the data using these indices
    transformed_data_sliced = transformed_data[:, target_var_index,
region_indices_reshaped[0], region_indices_reshaped[1]]
```

After this I returned previous form to the array:

```
data = []
for i in transformed_data_sliced:
    buf = i.flatten()
    data.append(buf)
data = np.array(data)
data = data.reshape(1646, 7, 21)
```

Then I started preparing data for the neural network. First of all I needed to expand dimensions of the data array:

```
data = np.expand_dims(data, 1)
```

After this I created X and Y: X is the current day, Y is the next day. After that I splitted X and Y into the training and test parts/

```
X = []
Y = []

for i in range(0, N-1):
  X.append(data[i])
  Y.append(data[i+1])

X = np.array(X)
Y = np.array(Y)
# Split on test train
X_train, Y_train = X[:int(0.8*N)], Y[:int(0.8*N)]
X_test, Y_test = X[int(0.8*N):], Y[int(0.8*N):]
```

To increase speed of learning I created batches:

```
    train_dataloader  =  torch.utils.data.DataLoader(list(zip(X_train,  Y_train)),
batch_size=16)
    test_dataloader   =   torch.utils.data.DataLoader(list(zip(X_test,   Y_test)),
batch_size=16)
```

4.4. Output data structure

In the output of this model I will get tensor of predicted values in the shape (7, 21)

4.5. Convolutional network with 4 layers

To build a model, I will use the method nn.Sequential from torch library. This method allows to combine different blocks of different neural networks, in that way it will be possible to create.

To start building a neural network, we need to create a baseline - simple model, that I will start from. So, I created a multilayer perceptron [11], with 5 layers. In this model I predicted 8 days by 7 previous days. X is 7 days, Y is 8 days, such pairs are built through the whole data array. Code of the model:

```
model = torch.nn.Sequential(
          torch.nn.Linear(21*7, 512),
          torch.nn.LeakyReLU(0.1),
          torch.nn.Linear(512, 256),
          torch.nn.LeakyReLU(0.1),
          torch.nn.Linear(256, 21*7)
        )


optimizer = torch.optim.Adam(model.parameters(), lr=0.00001)
```

This model goes through the 10 training epochs and is optimized by upper optimizer. In the result I got wrong output, because perceptron works badly with spatial data. So the next step was building a convolutional neural network[12].

To the input this model takes values from 4.3. Algorithm is the same as I illustrated before. Here is model structure:

```
model = torch.nn.Sequential(
          torch.nn.Conv2d(in_channels=1, out_channels=1,
                       kernel_size=(2,2), stride=1),
          torch.nn.Conv2d(in_channels=1, out_channels=1,
                       kernel_size=(3,3), stride=1),
          torch.nn.Flatten(start_dim=1),
          torch.nn.Linear(72, 21*7)
        ).to(device)
```

4.6. Neural network with 5 layers

To have more variants and better prediction function, I created one more model:

```
model2 = torch.nn.Sequential(
          torch.nn.Conv2d(in_channels=1, out_channels=1,
                       kernel_size=(4, 4), stride=1),
          torch.nn.Conv2d(in_channels=1, out_channels=1,
                       kernel_size=(2,2), stride=1),
          torch.nn.Flatten(start_dim=1),
          torch.nn.Linear(51, 64),
          torch.nn.Linear(64, 21*7)
        ).to(device)
```

This model also goes through 10 epochs and optimized by Adam[13], loss function is MSELoss[14]

5. Experimental evaluation

5.1. Metrics for experimental evaluation

For evaluating the effectiveness of the model and choosing the best one, I will use such metrics as MSE on the test part and relation between MSE and training epochs.

5.2 Numerical results of experimental evaluation

Here’s code for training model and MSE calculation:

```
loss_func = torch.nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.0001)
for epoch in range(10):
  epoch_loss_train = 0.
  epoch_loss_test = 0.

  # Training
  for x, y in train_dataloader:

    # Translate to GPU
    x = x.to(device)
    y = y.to(device)

    # Make prediction
    y_pred = model(x)

    # Find loss
    loss_value = loss_func(y_pred, y.flatten(1))

    # Calculate Gradient
    loss_value.backward()

    # Updating weights of the models
    optimizer.step()

    # Save value of loss on this epoch
    epoch_loss_train += loss_value.item()
# Validation
  for x, y in test_dataloader:
    x = x.to(device)
    y = y.to(device)
    y_pred = model(x)
    loss_value = loss_func(y_pred, y.flatten(1))


    epoch_loss_test += loss_value.item()

  # Print loss
  print(epoch_loss_train/len(train_dataloader),
epoch_loss_test/len(test_dataloader))
```

From this code we get that the best value of MSE for the first model is 0.004, for the second model the best value is 0.01. Let’s see models result on test part

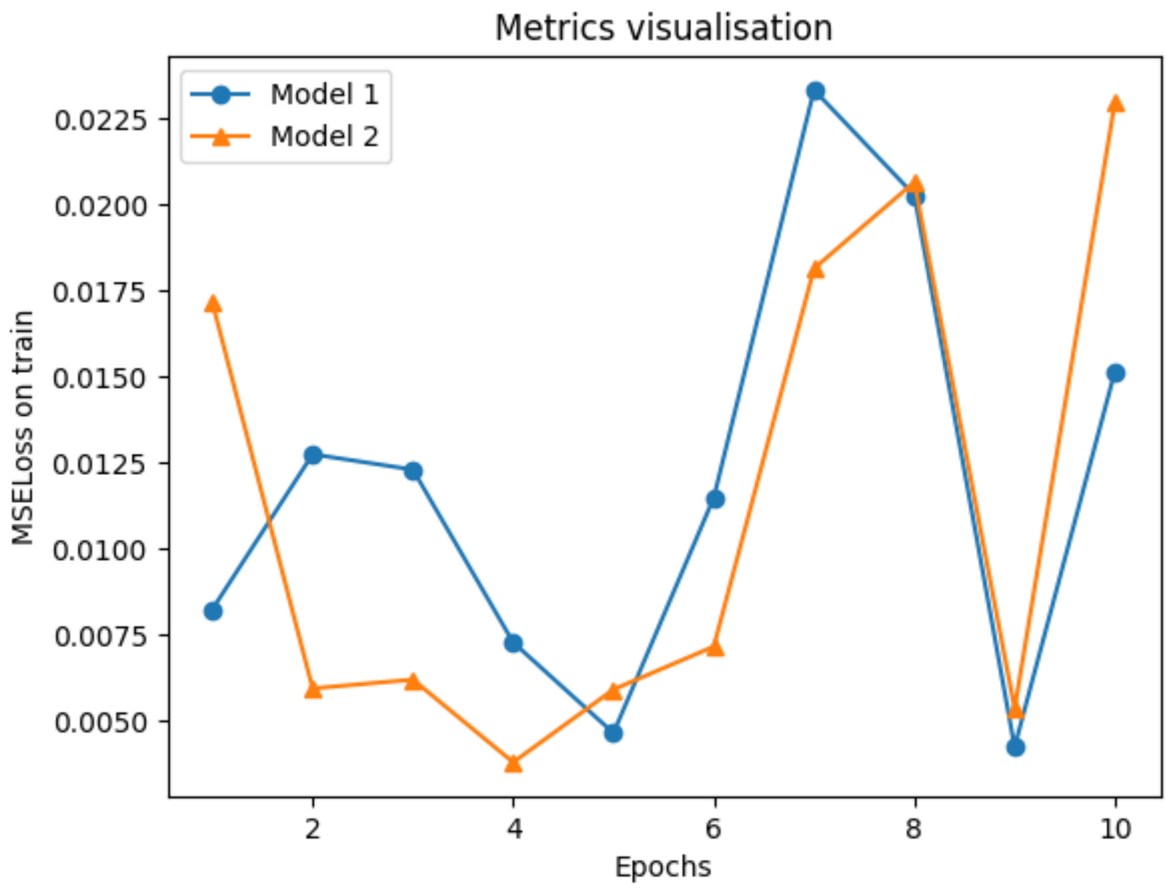


Figure 5.1 - Model’s result on test part

On this graph, we can see, that first model is training faster

As we can see from this factors, second model is more consistent, so I will use it to predict results and assess greenhouse emission

Let’s visualate our predicted values (figure 5.2). Application which visualates values, has problem with representing very small numbers, so the leftmost value on scale colorbar is 0, and the rightmost value is 0,000767.

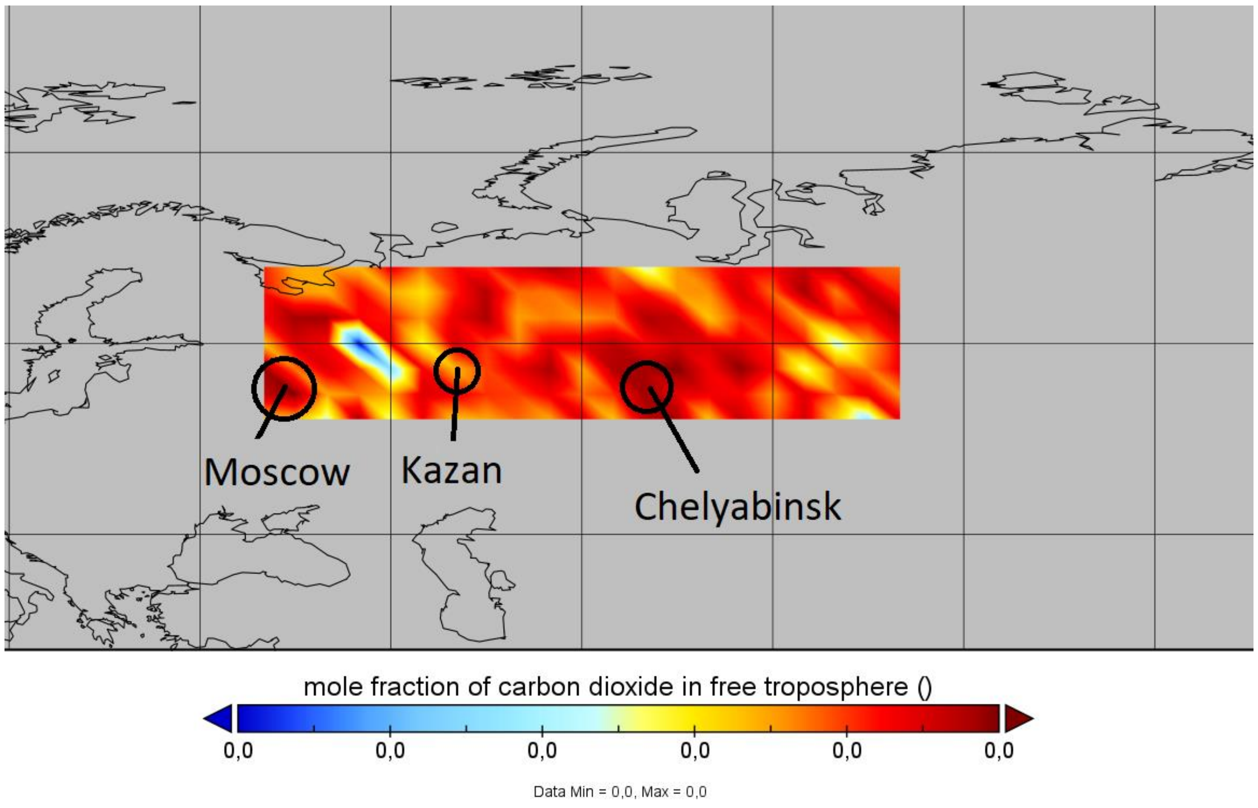

Moscow
Kazan
Chelyabinsk
mole fraction of carbon dioxide in free troposphere ()
0,0
0,0
0,0
0,0
0,0
0,0
Data Min = 0,0, Max = 0,0

Figure 5.2. - Predicted values visualization

5.3. Confirmed hypotheses

Let's use obtained result to check my hypothesis:

1. CO2 emission is high in the industrial zones: as we can see from this map, the biggest emissions are in Moscow and Urals - biggest industrial zones in Russia
2. The biggest emission is around heavy production: The biggest value in this map is located around Chelyabinsk - city with very heavy production, so it's true

5.4. Unsubstantiated hypotheses

Here is unsubstantiated hypotheses:

1. Cities with a population of more than a million people produce as much as in the big industrial zones: this hypothesis is wrong, as in Kazan, which has more than million people, emission is not so big.
2. In the forests CO2 emissions are small: as we can see, on the west part of our sample, where the forests located, emission is still big

5.5. Conclusions on the section

In the result of the neural network I got an MSE of 0,004 for scaled data. That means that our predicted values will differ from real values by 0.063, for unscaled this value equals 0.0001, which is a very good mark. That means my model has very good predictive power. So we can say about the nice quality of the model and consistency of the model.

Also the model is training very well, what we can see from its training graph, as it trains very fast.

Because of this metrics, I think that model gives consistent results and quality is good enough to take into account its predictions

Conclusion

High emission of greenhouse gasses is one of the biggest problems of the modern world, as it affects ecology and the health of humanity. It’s very important to learn this theme and assess it’s emissions, as it will allow us to find solution to solve this problem

During this project I have learned machine learning and neural networks. I have assessed greenhouse emission using different sources of information and applied different methods to evaluate and predict future emission.

To do this project I used different machine learning methods. First of all I built Time Series and Random Forest regressor machine learning models to predict future emissions for different categories of greenhouse emission. After that I got an array of future emissions and visualated it.

After that, I built architecture for a neural network to assess CO2 emissions using data from satellites. My architecture includes convolutional and linear blocks, to work with spatial data.

In the result of my work I got two working instruments of assessing greenhouse emission, made visualizations of obtained results, I found downwards trends of CO2, which is very hopefully

In the future, it’s possible to add to the model some new variables, such as other types of greenhouse emission, movements of air masses.

List of resources

[1]https://www.earthdata.nasa.gov/learn/backgrounders/remote-sensing#:~:text=Remote%20sensing%20is%20the%20acquiring,record%20reflected%20or%20emitted%20energy.

[2] Hastie, T.; Tibshirani, R.; Friedman, J. H. (2009). "10. Boosting and Additive Trees". The Elements of Statistical Learning (2nd ed.). New York: Springer. pp. 337–384. ISBN 978-0-387-84857-0. Archived from the original on 2009-11-10.

[3] https://en.wikipedia.org/wiki/Ordinary_least_squares

[4]Cohen, J., Cohen P., West, S.G., & Aiken, L.S. (2003). Applied multiple regression/correlation analysis for the behavioral sciences. (2nd ed.) Hillsdale, NJ: Lawrence Erlbaum Associates

[5]R.J.Oosterbaan, 1994, Frequency and Regression Analysis. In: H.P.Ritzema (ed.), Drainage Principles and Applications, Publ. 16, pp. 175-224, International Institute for Land Reclamation and Improvement (ILRI), Wageningen, The Netherlands. ISBN 90-70754-33-9 .

[6] E. GARİP and A. B. OKTAY, "Forecasting CO2 Emission with Machine Learning Methods," 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), Malatya, Turkey, 2018, pp. 1-4, doi: 10.1109/IDAP.2018.8620767.

[7] Gallo, Crescenzio & Contò, Francesco & Fiore, Mariantonietta. (2014). A Neural Network Model for Forecasting CO2 Emission. Agris on-line Papers in Economics and Informatics. VI.

[8] Ming Fan, Dan Lu, Siyan Liu, A deep learning-based direct forecasting of CO2 plume migration, Geoenergy Science and Engineering, Volume 221, 2023, 211363, ISSN 2949-8910

[9] https://facebook.github.io/prophet/

[10]https://catalog.data.gov/dataset/airs-aqua-l3-daily-co2-in-the-free-troposphere-airs-only-2-5-degrees-x-2-degrees-v005-airs

[11] https://en.wikipedia.org/wiki/Perceptron

[12] https://en.wikipedia.org/wiki/Convolutional_neural_network

[13] https://optimization.cbe.cornell.edu/index.php?title=Adam

[14] https://en.wikipedia.org/wiki/Mean_squared_error

[15] https://link.springer.com/referenceworkentry/10.1007/978-0-387-39940-9_803

[16] https://en.wikipedia.org/wiki/Time_series

[17] https://en.wikipedia.org/wiki/Interpolation

[18]https://en.wikipedia.org/wiki/Random_forest

[19]https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35

[20]https://programmathically.com/understanding-convolutional-filters-and-convolutional-kernels/