Predictive Modeling

In order to research and solve the question we put forward in the introduction, machine learning techniques are used to analyze and build a model that can adequately simulate real situations and provide a relatively reliable prediction for the future. In this project, linear regression and neural network are two elected techniques based on the existing dataset.

According to plots of correlation, every independent variable has a pretty distinct positive or negative relationship with the dependent variable. Besides, real-life experience can provide evidence that supports using a linear regression model. The neural network is chosen because of its complexity and adaptivity. The predicted result of different models can be used to compare, validate and analyze the advantages and disadvantages of different technologies and solve the question better.

In the introduction part, we decide to solve the problem of predicting CO_2 emission tendency per year. Therefore, the overall dataset is used for training in this project, while annual summary data is used for validating and testing.

Linear regression Modeling

Linear regression is a linear approach for modeling the relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables) (Freedman, 2009). In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. The most straightforward formulation of the predicted model is shown below.

$$f(x) = \beta_1 * x_1 + \beta_2 * x_2 + \cdots + \beta_p * x_p + \beta_0$$

In this project, since the training dataset has eight independent variables (exclude year), There will have nine parameters to fit the dependent variable. In the training process, mean squared error is a simple but effective way to determine the difference between model predictions and actual observations. The optimizer of this machine learning process is gradient descent to return the value that minimizes the result. The overall code is listed below.

a)
```
model (generic function with 1 method)
function model(x::Matrix{T}, p::Vector{T})::Vector{T} where T<:AbstractFloat
    return y = x * p[1:(length(p)-1)] .+ p[length(p)]
end
```

b)
```
mse (generic function with 1 method)
function mse(ŷ::Vector{T}, y::Vector{T})::T where T<:AbstractFloat
    mse = ((ŷ .- y).^2) ./ length(y)
    return sum(mse)
end
```

c)
```
minimize! (generic function with 1 method)
function minimize!(f_model::Function, x::Matrix{T}, y::Vector{T}, p::Vector{T}, η::T,
num_steps::Int)::Vector{T} where T<:AbstractFloat
    err(p) = mse(f_model(x, p), y)
    for num in 1:num_steps
        p -= err'(p) * η
    end
    return p

end
```

d)
```
train_model (generic function with 1 method)
function train_model(x::Matrix{T},y::Vector{T})::Vector{T} where T<:AbstractFloat
    b = Matrix(x)
    a = size(b)
    p = rand(a[2]+1)
    m = minimize!(model,b,y,p,0.01,10000)
    return m
end
```

Figure 1. Origin code for the linear regression model. a) shows the formula of the model. b) is the code for the model error. c) is the part of gradient descent. The learning rate of 0.01 and learning step 10000 are determined by the result of training. d) is the execution step to get the parameters of the linear regression model.

In Table 1, we can find the parameters which be calculated after the overall dataset is put in. The predicted result for the training dataset is shown in Fig 2. From Fig 2, we can find that this model works well on the training model. The error between the predicted and actual amounts is low enough to provide a solid prediction result. In the plot, we can find that 20 points are floating on the yellow linear, representing that the predicted amounts are much higher than the actual amount. This problem exists no matter how the learning rate and learning steps change. As a result, These data can be judged as incorrect data caused by inaccurate statistics.

Table 1. The parameters of linear regression model

β	1	2	3	4	5	6	7	8	0
value	92.2915	47.6183	61.2242	65.2719	46.2423	100.509	22.1533	101.883	0.801103

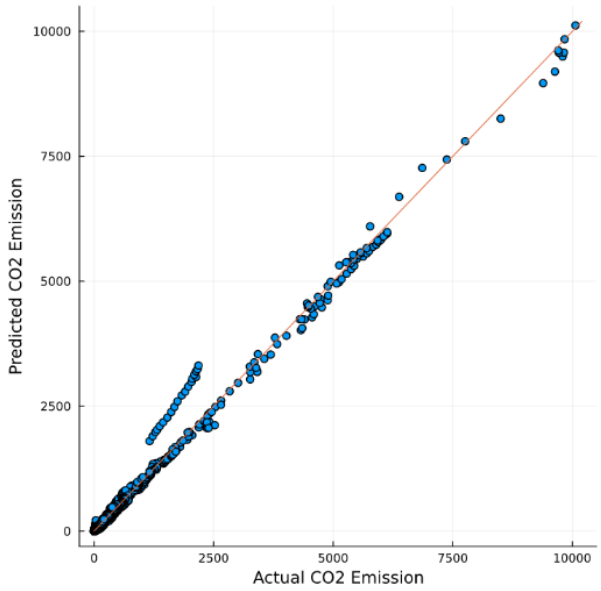

Figure 2. The result of applying the parameters on training data. The RMSE of this model on the training dataset is 73.3 kilotons. In the plot, the x-axis is the actual CO_2 emission amount, while the y-axis represents the predicted CO_2 emission amount calculated by the linear regression model. The yellow represents that the predicted amount is equal to the actual amount.

Validation the linear regression Model

In order to affirm whether the parameters can fit appropriately in other conditions, testing datasets are introduced to check the veracity of this model. To guarantee the complexity of the testing dataset, There are four training datasets, the world's annual CO_2 emission, The United States' annual CO_2 emission, China's annual CO_2 emission, and Russia's annual CO_2 emission. The three countries' data are chosen since these three countries are the first three countries in the rank of total CO_2 emission. The result of validation is shown in Fig 3.

Fig 3 indicates that the linear regression model works well in most cases. The predicted results are similar to actual data in most cases. However, there are still several errors that are demonstrated in the plot.

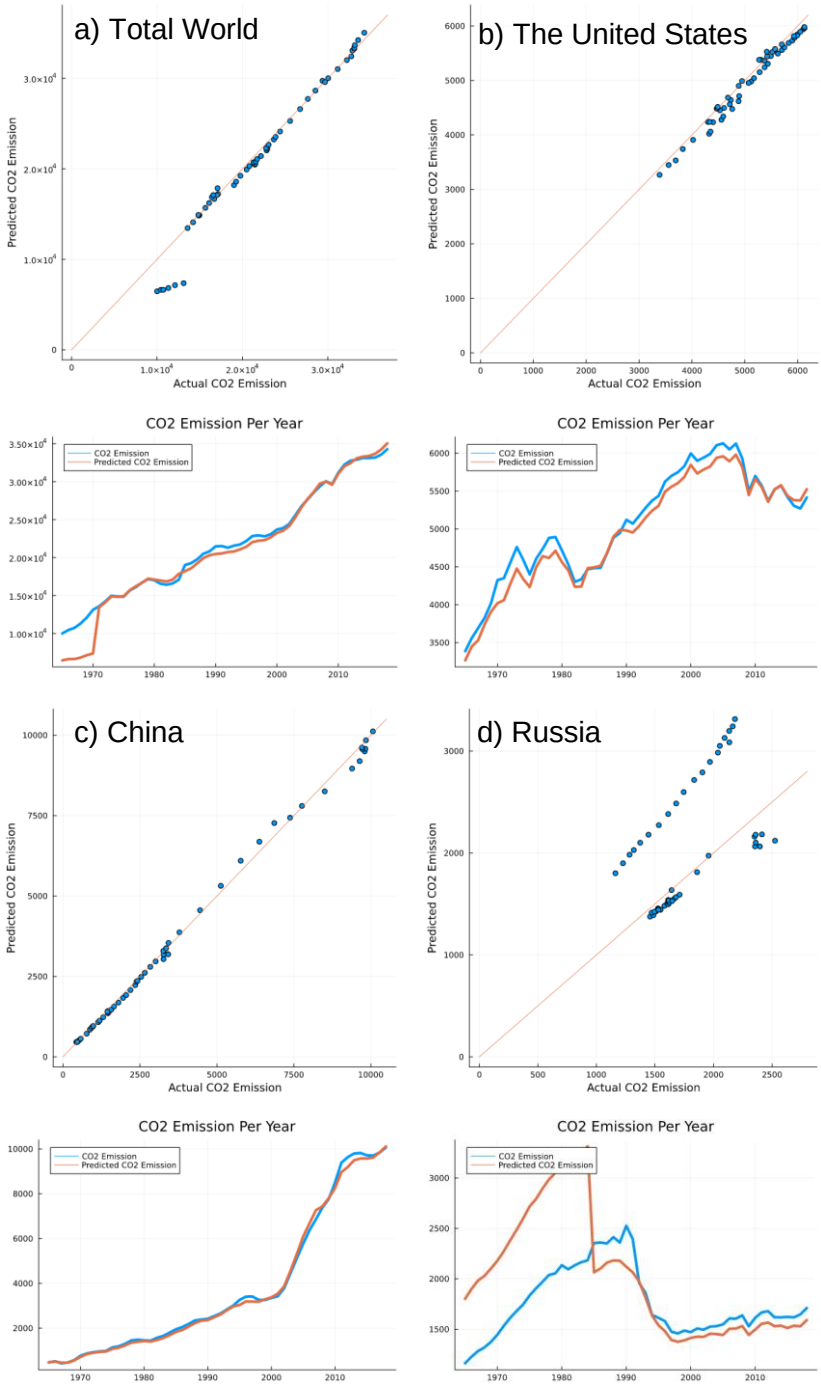

Figure 3. The results of applying linear regression model on testing datasets. The upper plot in each section shows the data difference between the predicted and actual amounts. The bottom plot indicates CO_2 emission data changing over time change. The yellow line represents the predicted amount, while the blue line represents the actual amount. a) the result of applying the model to annual CO_2 emission data. The RMSE is 1569.8 kilotons. b) the result of applying the model to The United States CO_2 emission data. The RMSE is 143.2 kilotons. c) the result of applying the

model to China CO_2 emission data. The RMSE is 159.0 kilotons. d) the result of applying the model to Russia CO_2 emission data. The RMSE is 545.2 kilotons.

In forecasting results for total world CO_2 emission amount, six points are remarkably lower than actual amounts. From the CO_2 emission and year relation plot, we can find these points fasten on the first several years in the timeline. After checking the original dataset, we find that the reason behind this phenomenon is that there is no statistical data for oil consumption from 1965 to 1970. Since oil consumption plays a significant role in CO_2 emission, the missing data can easily cause mistakes in data forecasting.

The other significant fault is shown in Russia's CO2 emission dataset. From the bottom plot, we can find the imitative effect of the training model keeps a deficient level from 1965 to 1985. After that time, the prediction becomes more accurate. As I mentioned before, there are 20 points that contain incorrect data. After comparing the original dataset, we find that these data have a statistical problem. The tendency is strange, which is shown in Fig 4.

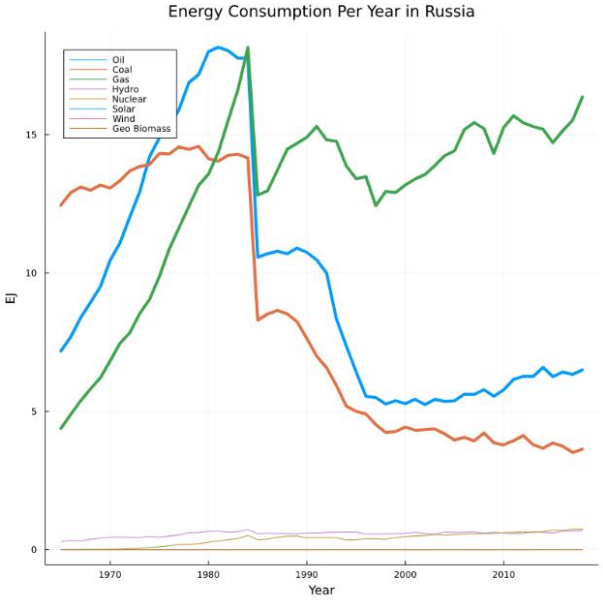

Figure 4. Different kinds of energy consumption per year in Russia. In the plot, different species of energy consumption ways are separated by color.

In general, the Linear regression model serves properly to predict CO2 emission in different countries when all of the independent variables all in readiness. This model can be used to estimate CO2 emissions after acquiring energy consumption data. In the real world, the government calculates CO2 emissions with the guidance of the Intergovernmental Panel on Climate Change (IPCC). The fingerpost from IPCC only concentrates on oil, coal, and gas, which is reasonable in the real world. The achievement of this project might play a complementary role in estimating CO2 emissions in every country.

Neural Network Modeling

The second model for prediction is the neural network model. A neural network is a network or circuit of biological neurons or, in a modern sense, an artificial neural network composed of artificial neurons or nodes (Hopfield, 1982).

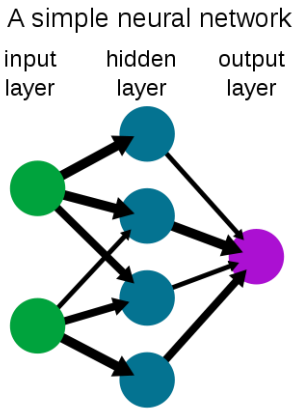

Figure 5. Deep neural network schematic diagram

The difference between the neural network models and linear regression is the size of the output. Linear regression returns a $1 \times o$ matrix, while a neutral network returns a $h \times o$ matrix. Since that, It is practicable and meaningful to compare the result of a different model. Theoretically, a proper neural network model can provide us with a better consequence since the parameters' amount is much higher and more complex. As a result, we build a neural network model for our dataset. The training and test dataset is the same one as the linear regression model to obtain a better comparison.

In this project, the modeling part is a three-layer neural network. Although the layer amount is not that high, it works pretty well in the training dataset. Root-mean-square deviation works appropriately in neural network model training. The optimizer of this machine learning process is gradient descent to return the value that minimizes the result. The overall code is listed below.

a)

```
dense (generic function with 1 method)
function dense(x,p)
    w,b = p
    w*x .+ b
end
```

```
nn_model (generic function with 1 method)
function nn_model(x,p)
    w1,b1,w2,b2,w3,b3, = p
    o1 = relu.(dense(x,[w1,b1]))
    o2 = relu.(dense(o1,[w2,b2]))
    dense(o2,[w3,b3])
end
```

b)

```
rmse (generic function with 1 method)
function rmse(ŷ::Vector{T}, y::Vector{T})::T where T<:AbstractFloat
    mse = ((ŷ .- y).^2) ./ length(y)
    return (sum(mse))^0.5
end
```

c)

```
train! (generic function with 1 method)
function train!(modelf, errf, p::AbstractVector, x::Matrix{T}, labels::Vector{T},
η::T, nsteps::Int) where T<:AbstractFloat

    e(p) = errf(modelf(x, p)[:], labels)

    for i in 1:nsteps
        g = e'(p)
        p .-= η .* g
    end
    return p
end
```

d)

```
normalize_df (generic function with 1 method)
function normalize_df(x_train::Matrix)
    xm = mean(Matrix(x_train),dims=1)
    xs = std(Matrix(x_train),dims=1)
    x = (Matrix(x_train) .- xm) ./xs
    F = svd(x')
    x = (F.U * x')'
    return x
end
```

e)

```
begin

    x1 = normalize_df(x)
    inputsize = size(x1,2)
    hiddensize = 500
    nlabels = 1


    w1 = rand(hiddensize, inputsize) .- 0.5
    b1 = rand(hiddensize) .- 0.5

    w2 = rand(hiddensize, hiddensize) .- 0.5
    b2 = rand(hiddensize) .- 0.5

    w3 = rand(nlabels, hiddensize) .- 0.5
    b3 = rand(nlabels) .- 0.5

    p = [w1,b1,w2,b2,w3,b3]

    η = 0.001
    nsteps = 1000

    p = train!(nn_model, rmse,
        p, x1', y, η, nsteps)
end
```

Figure 6. Origin code for the neural network model. a) shows the formula of the three-layer model. b) is the code for the model error. c) is the part of gradient descent. The learning rate of 0.001 and learning step 1000 are determined by the result of training. d) is the part for data normalization. e) is the execution step to generate and optimize initial parameters for the model. Batch size of this model is 500, which is large enough to provide favorable result.

The predicted result for the training dataset is shown in Fig 7. In comparison between the linear regression model and neural network model, root-mean-square deviation decreased by 30.96%, which is tremendous progress in consideration of the linear regression model predict accurately. In this plot, 20 points higher are than the baseline, which are the data points of Russia from 1965 to 1985, as the report mentioned before. This phenomenon strengthens the speculation that a statistical mistake happened at that time. Therefore, neither model can handle this data. In general, the neural network model performs better on the training dataset than the linear regression model.

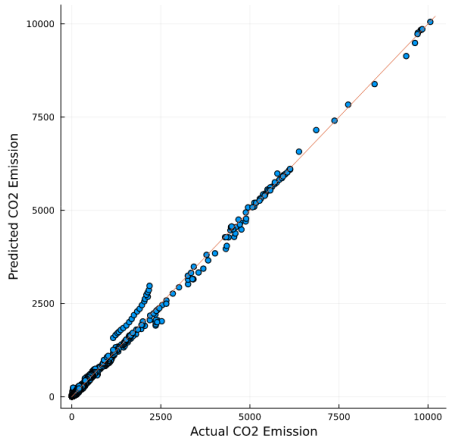

Figure 7. The result of applying the parameters on training data. The RMSE of this model on the training dataset is 50.6 kilotons. In the plot, the x-axis is the actual CO_2 emission amount, while the y-axis represents the predicted CO_2 emission amount calculated by the linear regression model. The yellow represents that the predicted amount is equal to the actual amount.

Validation the Neural Network Model

Fig 8 indicates that the linear regression model works terribly in cases other than the training dataset. The predicted results are totally different from the actual data in most cases. Compared to the linear regression model, although the neural network model performs better in the training dataset, it has no practical value since it fails in the test datasets.

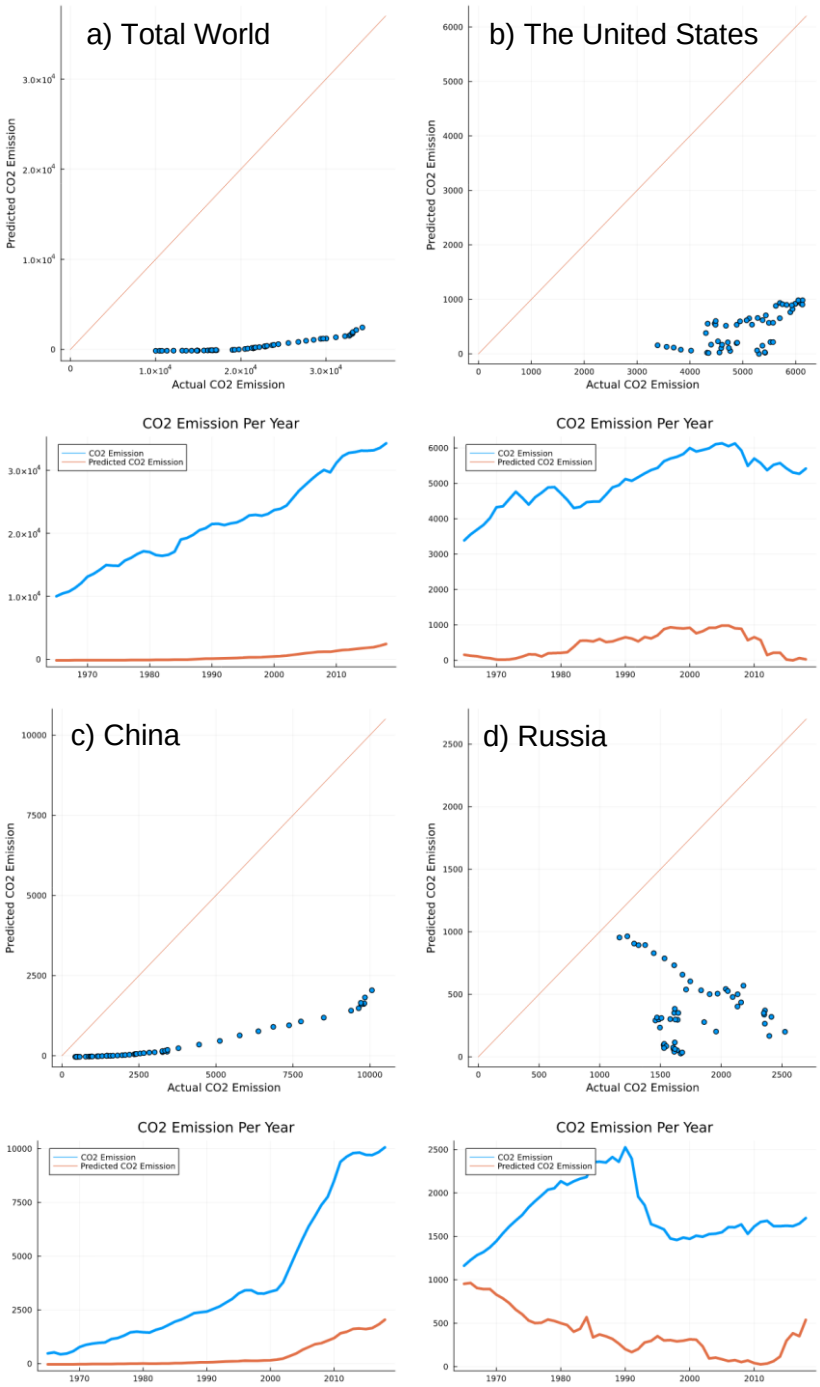

Figure 8. The results of applying the neutral network model on testing datasets. The upper plot in each section shows the data difference between the predicted and actual amounts. The bottom plot indicates CO_2 emission data changing over time change. The yellow line represents the predicted amount, while the blue line represents the actual amount. a) the result of applying the model to annual CO_2 emission data. The RMSE is 22240.4 kilotons. b) the result of applying the model to The United States CO_2 emission data. The RMSE is 4648.1 kilotons. c) the result of applying the model to China CO_2 emission data. The RMSE is 4224.8 kilotons. d) the result of applying the model to Russia CO_2 emission data. The RMSE is 1451.5 kilotons.

Reference

Freedman, D. A. (2009). Statistical Models: Theory and Practice (2nd ed.). Cambridge University Press.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences, 79(8), 2554–2558. https://doi.org/10.1073/pnas.79.8.2554

Re Cecconi, F., Moretti, N., & Tagliabue, L. (2019). Application of artificial neutral network and geographic information system to evaluate retrofit potential in public school buildings. Renewable and Sustainable Energy Reviews, 110, 266–277. https://doi.org/10.1016/j.rser.2019.04.073