Unveiling the Future of Carbon Emissions Trading: A Machine Learning and Neural Network Perspective on Regional Markets.

GROUP - 07

Prayag Nikul Purani (017416737)

Department of Applied Data Science

San Jose State University

DATA 270: Data Analytics Processes

Submitted to: Dr. Eduardo Chan

May 17, 2024

Introduction

The period since the 1950s, known as the "atomic age of science and technology," has witnessed remarkable advancements alongside growing concerns regarding carbon emissions and environmental impact. The COVID-19 pandemic in 2019 led to a notable 5.4% reduction in carbon emissions in 2020, prompting reflection on the interconnectedness between human activities and environmental consequences. Carbon dioxide, primarily emitted from vehicles and industries, remains a significant contributor to greenhouse gas emissions and climate change. Research highlights the carbon footprint associated with training large-scale machine learning models, emphasizing the environmental impact of technological innovation. Over the past five decades, significant shifts in environmental consciousness and policy initiatives have occurred, including international agreements like the Kyoto Protocol and the Paris Agreement. These agreements have shaped global efforts to combat climate change and regulate carbon emissions. The predictive modeling approach adopted by the project aims to forecast future carbon dioxide emissions based on historical data, policy interventions, economic trends, and technological advancements. This multilabel prediction task involves predicting emissions across multiple sectors simultaneously, capturing the nuanced relationships between different factors influencing emissions. The dataset consists of 40 features derived from various sources, with the target feature being "Shared CO2," representing total carbon dioxide emissions across all sectors.

Through predictive modeling, the project seeks to generate insights into future emissions trends, assess the effectiveness of policy interventions, and inform decision-making processes for mitigating climate change and promoting sustainable development. The multilabel prediction scenario enables a comprehensive assessment of carbon emissions, facilitating targeted interventions and strategies for reducing overall carbon footprints.

Model Development

4.1 Model Proposals

Boateng et al. (2020) created models to anticipate China's carbon emissions from buildings based on five predictors: population, GDP, R&D investment, urbanization, and energy use. Using 140 observations, they examined six machine learning techniques, and 36 observations were used for validation. With a prediction accuracy of 99.88%, the random forest technique outperformed the other algorithms, K-nearest neighbors (KNN), XGBoost, decision trees, AdaBoost, and support vector regression (SVR). Even though random forest was the best performer, KNN also produced predictions that were precise and timely. The most time-efficient method was the decision tree. In order to provide prompt policy responses, the authors advise policymakers to use random forest, KNN, and decision tree models to provide accurate, real-time projections of China's construction emissions.

Ketkar et al., 2021 talked about how convolutional neural networks (CNN) are employed to extract spatial and temporal patterns from the data, which can be especially beneficial for image or sequence-based data. Finally, (Dongare et al., 2012) artificial neural networks (ANN) are used to learn patterns and relationships, offering flexibility and scalability in modeling various types of data. The project culminates in the implementation of a hybrid model combining CNN and ANN, leveraging the strengths of both architectures to achieve superior predictive performance. The (Sani & Gahwera, 2023) CNN-ANN hybrid model serves as the main predictive model for the project, integrating the insights and features learned from the preceding models to deliver robust and accurate predictions.

Overall, this progressive flow of modeling techniques allows the project to iteratively build upon previous approaches, leveraging the strengths of each method to improve prediction

accuracy and uncover nuanced patterns within the data. The hybrid CNN-ANN model represents the pinnacle of this approach, combining the best aspects of various modeling techniques to achieve optimal results.

Hybrid CNN-ANN

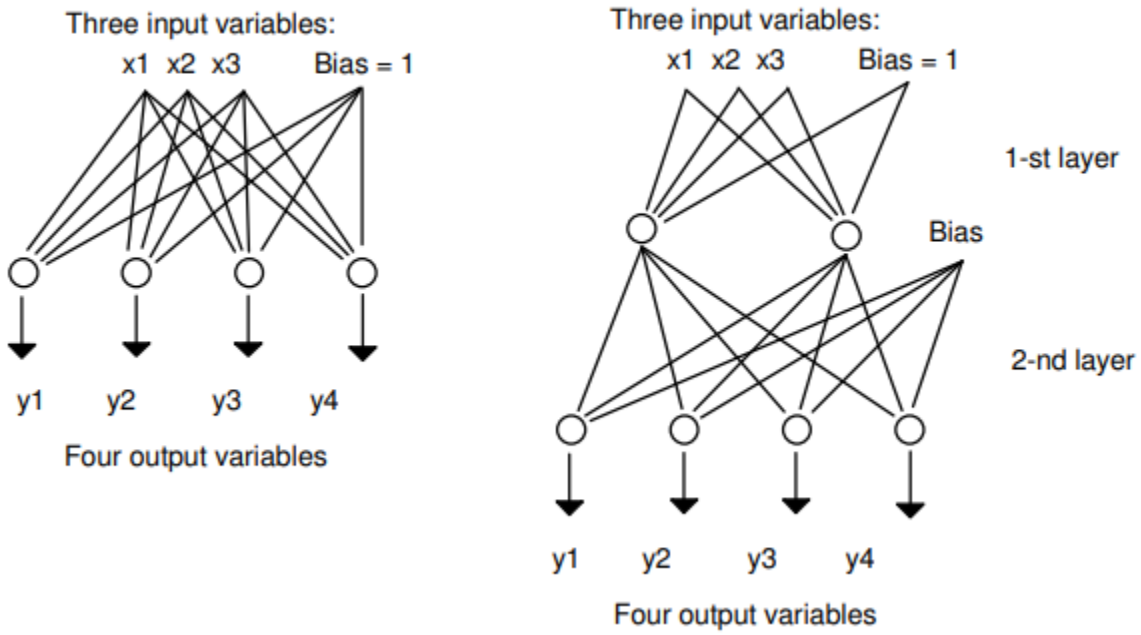
A hybrid CNN-ANN (Convolutional Neural Network-Artificial Neural Network) approach is employed to predict carbon emissions for future years using global data. This method combines the strengths of CNNs in extracting spatial and temporal patterns from large-scale datasets, such as satellite imagery or climate data, with the capabilities of ANNs in learning complex relationships between input features and target variables. The hybrid model utilizes deep learning and neural networks to enhance the precision and resilience of carbon emission forecasts. This assists policymakers and researchers in comprehending and addressing the consequences of climate change worldwide.

ANN (Artificial Neural Network). The concept of (Zupan, 1994) Artificial Neural Networks (ANNs) is indeed inspired by the neural found in the brain. The neurons are interconnected in complex networks, and they communicate with each other through electrical signals. Similarly, in artificial neural networks (ANNs), artificial neurons are linked together in layers, and information is processed through these interconnected layers. The analogy to the communication between the brain and muscles is particularly relevant in the context of feedforward neural networks, where information flows from input nodes through hidden layers to output nodes in a forward manner, without feedback loops. This is akin to the brain sending signals from the sensory organs to muscles for action without receiving direct feedback from the muscles (Zupan, 1994).

Figure 1 illustrates the architecture of both a one-layer network and a two-layer network. In the one-layer network, there are four neurons, each equipped with four weights, resulting in a total of 16 weights. These neurons receive input signals, along with an additional input from the bias, which remains constant at one. Despite the fixed bias input, the weights associated with the connections between the bias and the neurons remain adjustable. In contrast, the two-layer network has six neurons, divided into two layers: two in the initial layer and four in the subsequent (output) layer. Similar to the one-layer network, all neurons in a given layer receive signals from the layer directly above. In total, the two-layer network encompasses 20 weights, with eight located in the first layer and 12 in the second layer.

Figure 1

The depicted artificial neural networks (ANNs), with a one-layer structure on the left and a two-layer structure on the right, are designed to address a problem involving three variables as input and four possible responses as output.



Note. By, Zupan, 1994 basic one and two-layer ANN.

Table 1

Different types of ANN

Name	**Description**
Single-layer feed-forward network	Enables the one-way transmission of information, only from the input layer to the output layer, without any feedback loops.
Multi-layer feed-forward network	The system is composed of numerous layers of neurons, with information flowing in a unidirectional manner from input to output through hidden layers, without any feedback connections within the layers.
Single node with own feedback	A neural network configuration where a single node receives its own output as feedback, allowing it to consider its own past outputs during computation.
Single-layer recurrent network	A neural network architecture where feedback connections are present, allowing information to loop back within the same layer, enabling the network to process sequential data or time series.

Note. Different types of ANN

Table 2

The formula used for the ANN

Name	Formula	Description
Error back-propagation	δε = y - t	The exact error made by the ANN model at every node is calculated using this formula
Learning Error (momentum)	Δwji = ηδj yi + μΔwji	The parameters η and μ, commonly referred to as the learning rate and momentum, respectively, play crucial roles in training artificial neural networks. Typically, these parameters fall within the range of 0.1 to 0.9.
Learning error back-propagation	$ℏky¨=f((Wk*x)ij+bk)$	During each epoch, the network receives inputs, produces outputs, compares them with the target outputs, and adjusts its weights accordingly to minimize the error.
Sigmoid function	1 / (1 + exp(-x))	Mathematical function utilized to transform input values into a range (0.1), commonly employed in artificial neural networks.

Note. ANN formula

Table 3

ANN Algorithm

Algorithm for ANN
Define a Neural Network:

Create random initial weights and biases for connections between layers.

Train the Neural Network:

Provide input data and target labels.

For each training epoch:

Forward pass: Compute output using current weights and biases.

Calculate error for predicted and actual.

Backward pass: Adjust weights and biases to minimize error.

Repeat until convergence or a fixed number of epochs.

Optionally, monitor and print the loss.

Test the Trained Neural Network:

Provide new input data.

Forward pass: Compute output using trained weights and biases.

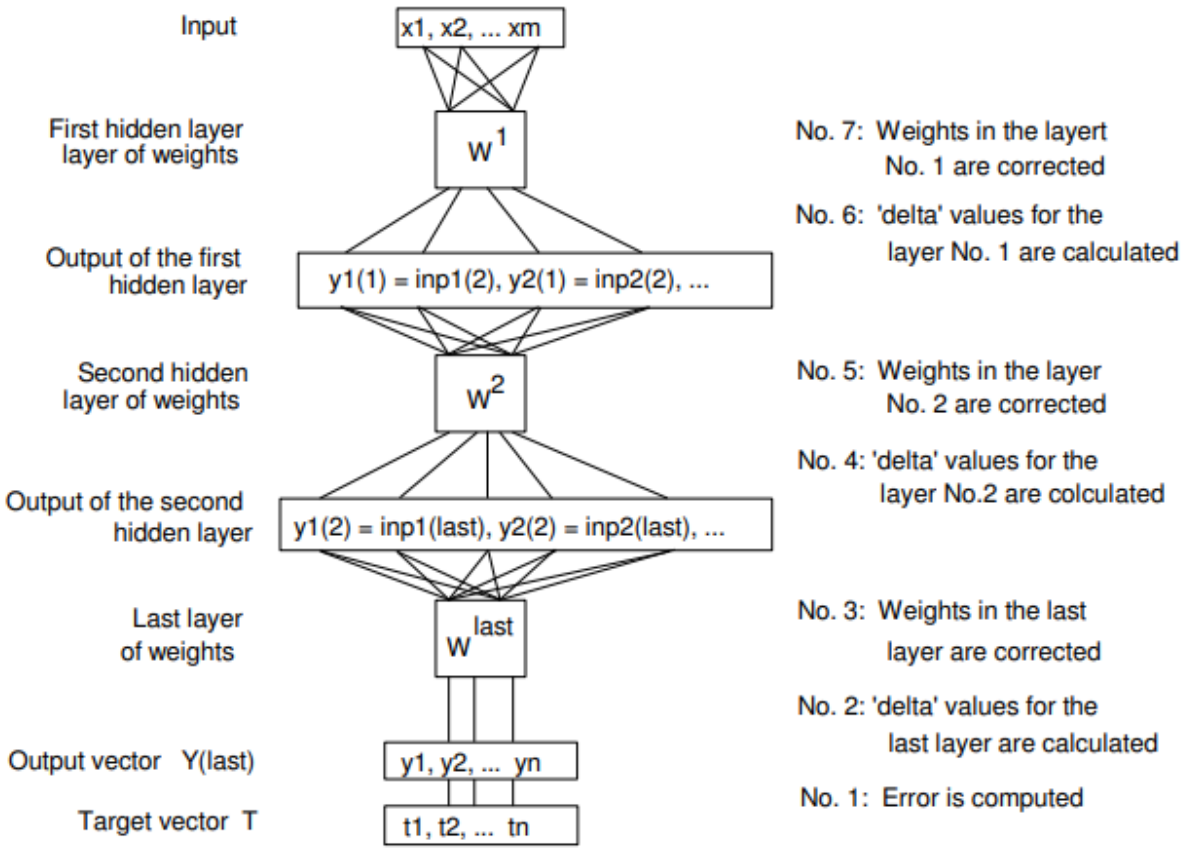
Get the predicted output.

Compare with the actual output.

Note. Algorithm for the ANN

Figure 2

ANN Architecture

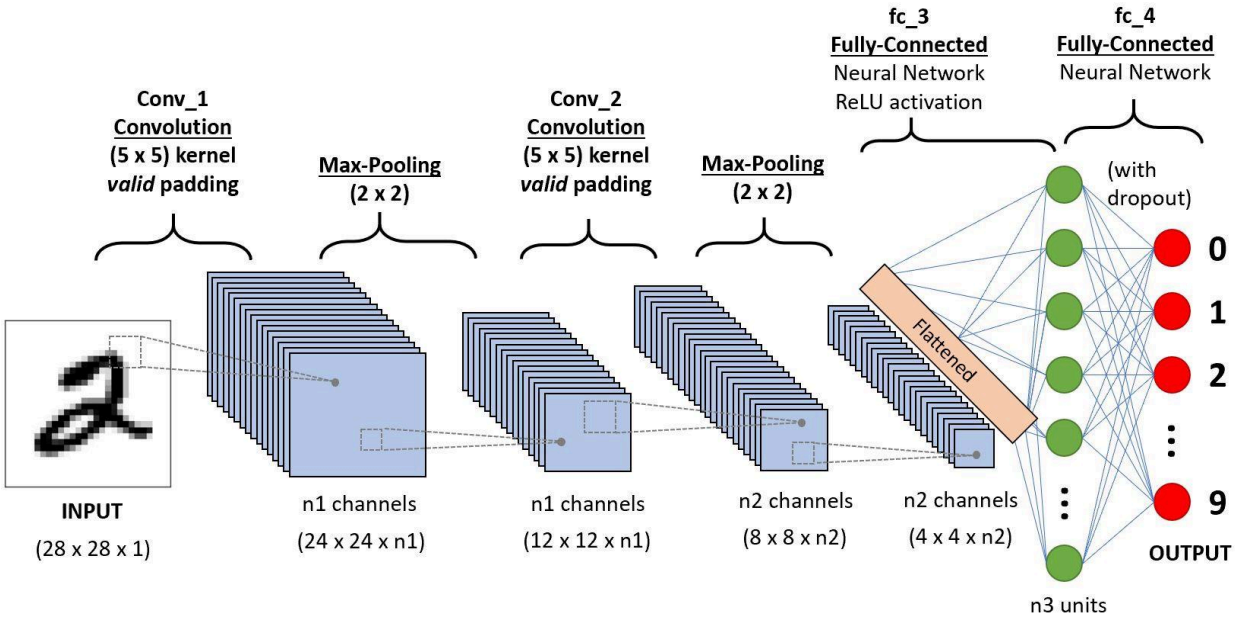


Note. By Zupan, 1994 for the ANN order correction of weights in error backpropagation.

Convolutional Neural Networks (CNN). are a category of deep learning models extensively utilized for analyzing data structured in a grid-like manner, such as images. They necessitate input data in a tensor format, featuring at least three dimensions: height, width, and channels. The data undergoes sequential processing through various layers, each dedicated to specific operations. These layers encompass the input layer, hidden layer(s), and output layer, collectively constituting the neural network architecture (Yamashita et al., 2018). Figure 3 provides a higher-level overview of this sequential processing.

Figure 3

CNN Architecture



Note. By Ratan, 2020 Convolutional Neural Network Architecture.

Table 4

Keywords in CNN

Name	**Description**
Pooling	Pooling involves taking a small portion of the input data and summarizing it by either averaging (average pooling) or selecting the maximum value (max pooling), providing a condensed representation of the input. This process allows for the extraction of key features while reducing the dimensionality of the data.
Activation function	The activation function serves as a node positioned at the end of or between neurons in neural networks. It plays a critical role in determining whether a neuron should be activated (or "fire") based on the input it

	receives, thereby used to get complex non-linear relationships from data.
Feature buffer	On-chip memories store output data of sub-blocks in neural networks. Each sub-block saves its activation map in separate memories, typically sized at 10k bits. If the computation depth exceeds 10k, additional memory components of the same size are utilized.
Padding	Padding with zeros is applied before model training to address the discrepancy between input and output matrix sizes and prevent data loss, particularly at the boundaries where information may be truncated.
Striding	Striding involves skipping certain neighboring cells during the data processing step, allowing for the visualization of data sub-sampling.

Note. Keywords are explained in the above table.

Table 4 shows the important keywords and the steps that are performed in CNN. Now, the table below holds the implementation formulas that built CNN from scratch and the specific description for each step. The following table x outlines the implementation formulas and detailed descriptions for each step in building a Convolutional Neural Network (CNN) layer and their implementation formula.

Table 5

CNN Formula for each step.

Name	**Formula**	**Description**
Convolution	$Y_{i,j}$	Let X X be the input tensor, W W be the filter/kernel, b b be the bias term, and Y Y be

	$=\sum_m\sum_n X_{i+m,j+n} \cdot W_{m,n}+ b$	the output feature map.
ReLU Activation	$A=\max(0,Z)$	To introduce non-linearity into the network, an activation function is applied element-by-element after the convolution process. Tanh, sigmoid, and ReLU (Rectified Linear Unit) are common activation functions.
Pooling (Downsampling)	$Y_{i,j}=\max_{m,n}(X_{i+m,j+n})$	Pooling layers reduce feature map spatial dimensions without sacrificing important information.
Fully Connected Layers	$Y=\text{Activation}(X \cdot W+b)$	Fully connected layers take the flattened output of the previous layers and perform matrix multiplication with a weight matrix followed by a bias term and an activation function. Let X X be the flattened input, W W be the weight matrix, b b be the bias term, and Y Y be the output.

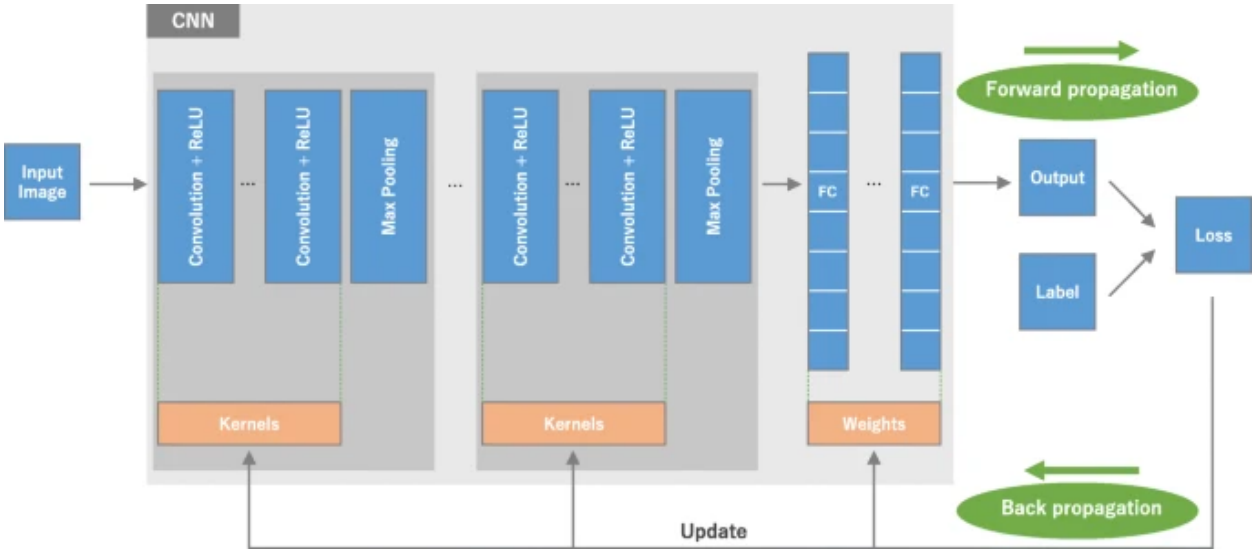
Note. CNN formula step vise

The CNN design is shown in Figure 5. It consists of several linked building pieces, including convolution layers, fully connected layers, and pooling layers. In the training phase, a loss function is applied via forward propagation on a training dataset to evaluate the model's

performance with certain kernels and weights. Learnable parameters—such as kernels and weights—are modified using backpropagation with a gradient descent optimization technique in accordance with the loss value. CNNs often use the Rectified Linear Unit (ReLU) activation function to add non-linearity and sparsity to the network.

Figure 4

CNN Architecture



Note. Rikiya et al, 2018 describe their research

Table 6

CNN Algorithm

CNN Algorithm
Inputs: - X: Input features (training data) y: Target outputs Input: Number of input features

Hidden: Number of neurons in the hidden layer

Output: Output neurons

Epochs: Training epochs

Learning rate: Learning rate for gradient descent

Initialize weights and biases randomly:

Define activation function: -

sigmoid(x) returns 1 / (1 + exp(-x))

Define derivative of activation function: -

sigmoid_derivative(x) returns x * (1 - x)

For each training epoch from 1 to epochs:

Forward pass

Backward pass

Update weights and biases

End loop

Output: Trained neural network with updated weights and biases

Note. Algorithm for CNN.

Hybrid CNN-ANN. The CNN part of the hybrid model excels at extracting meaningful features from complex data like images. By leveraging convolutional layers, pooling layers, and other components of the CNN architecture, the model can automatically learn to identify patterns and features at different levels of abstraction within the input data. These hierarchical representations capture important characteristics of the input, which are then passed on to the subsequent layers of the model for further processing. In essence, the CNN serves as the feature extractor, transforming raw input data into a representation that preserves relevant information

while discarding noise and irrelevant details. This extracted feature representation is then used by the subsequent layers of the model, typically an ANN, for making predictions or decisions tailored to the specific task at hand. After the CNN layers have extracted relevant features from the input data, typically represented as feature maps, the output from the last CNN layer needs to be flattened or reshaped into a vector format before it can be passed on to the fully connected layers of the ANN. The reason for this flattening or reshaping step is to ensure that the output from the CNN layers is compatible with the input requirements of the ANN layers. Fully connected layers in ANNs expect one-dimensional input vectors, where each element in the vector represents a feature or a combination of features extracted from the input data. Flattening the output essentially collapses the spatial dimensions of the feature maps into a single dimension, resulting in a one-dimensional vector. Alternatively, reshaping can also be used to rearrange the dimensions of the feature maps to match the input size expected by the fully connected layers of the ANN.

Table 7

Hybrid CNN-ANN

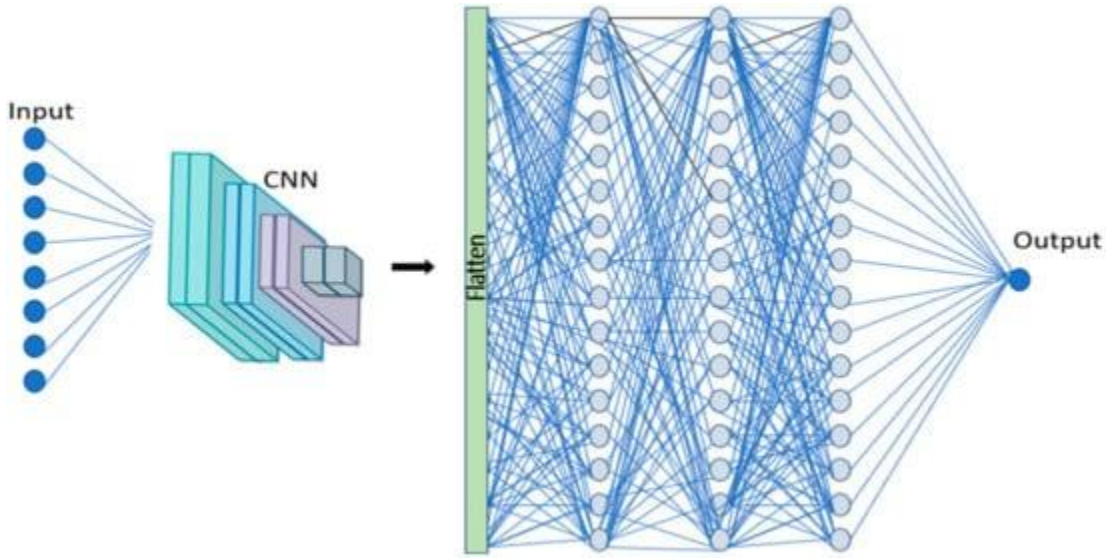
Algorithm
ConvolutionalLayer(image, filters)
MaxPoolingLayer(feature_maps, pool_size)
FullyConnectedLayer(input_vector, weights, biases)
HybridModel(input_image, cnn_filters, cnn_biases, cnn_pool_size, ann_weights, ann_biases)
feature_maps = ConvolutionalLayer(input_image, cnn_filters)
pooled_feature_maps = MaxPoolingLayer(feature_maps, cnn_pool_size)

```
        flattened_output = Flatten(pooled_feature_maps)
        output = FullyConnectedLayer(flattened_output, ann_weights, ann_biases)
        return output
Define cnn_filters, cnn_biases, cnn_pool_size, ann_weights, ann_biases
input_image = LoadInputImage()
output = HybridModel(input_image, cnn_filters, cnn_biases, cnn_pool_size, ann_weights,
ann_biases)
Display(output)
```

Note. Hybrid CNN-ANN algorithm.

Figure 5

Hybrid CNN-ANN.



Note. According to Mukhtar et al, 2022 hybrid CNN-ANN.

4.2 Model Supports

Environment, Platform, and Tools

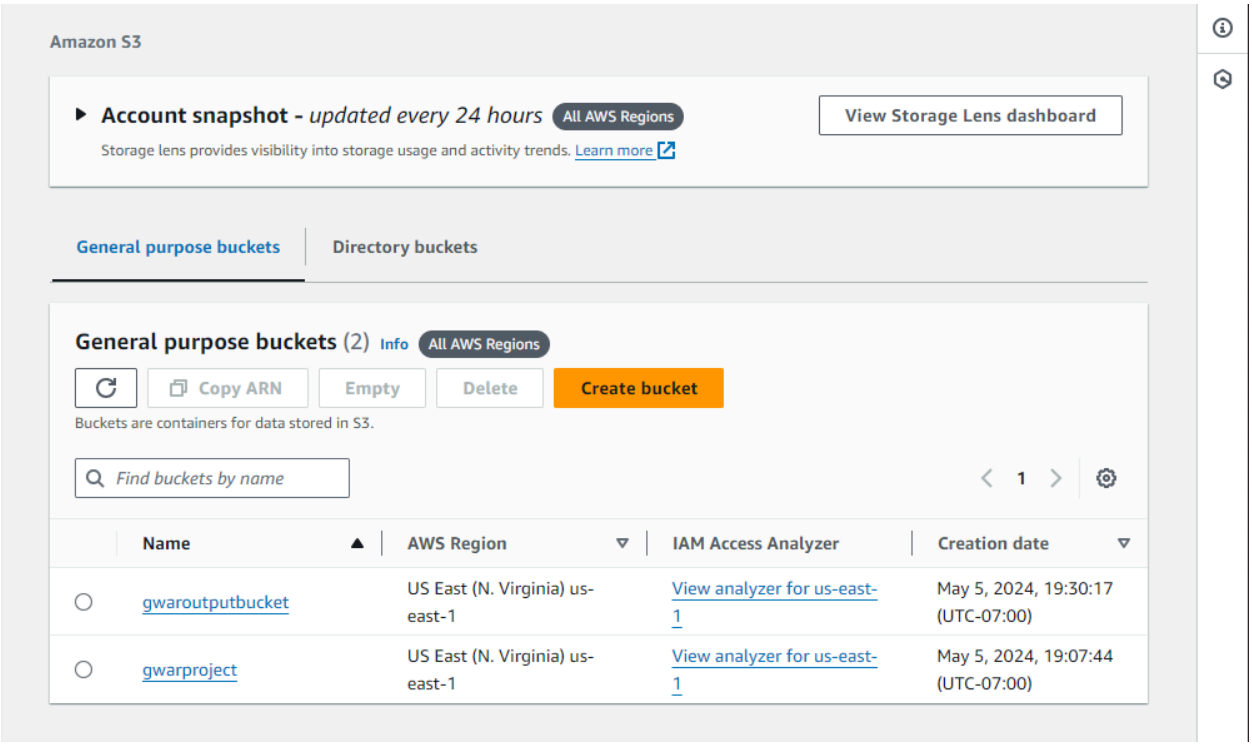
Creating a separate environment in Anaconda Navigator for projects involving libraries like TensorFlow and Keras is advisable to avoid conflicts with existing dependencies. This isolation ensures smooth management of specific versions and dependencies, safeguarding other projects from potential issues. Using Anaconda Navigator's GUI simplifies the process. Users navigate to the "Environments" section, click "Create," and specify the environment's name and Python version. They can also select additional packages as needed. This user-friendly approach streamlines environment setup, promoting compatibility and organization across projects.

With the Anaconda environment now set up, attention shifts to configuring the necessary infrastructure on Amazon Web Services (AWS) to facilitate data storage, processing, and analysis. The key components of this AWS setup include S3 buckets for data storage, AWS Glue crawlers for the Extract, Transform, Load (ETL) process, and AWS Athena for querying and analyzing the data stored in the data warehouse.

Setting up the AWS infrastructure for the project begins with creating S3 buckets via the AWS Management Console. Two main buckets are created: one for data storage and another for storing Athena query outputs. The data storage bucket acts as a central repository for datasets and model artifacts, offering scalability, durability, and accessibility. Storing data in S3 ensures high availability and reliability for processing and analysis. The second bucket is dedicated to storing Athena query results, facilitating better organization and management of insights. Separating input and output data maintains data integrity and clarity in the workflow.

Figure 6

S3 buckets formation

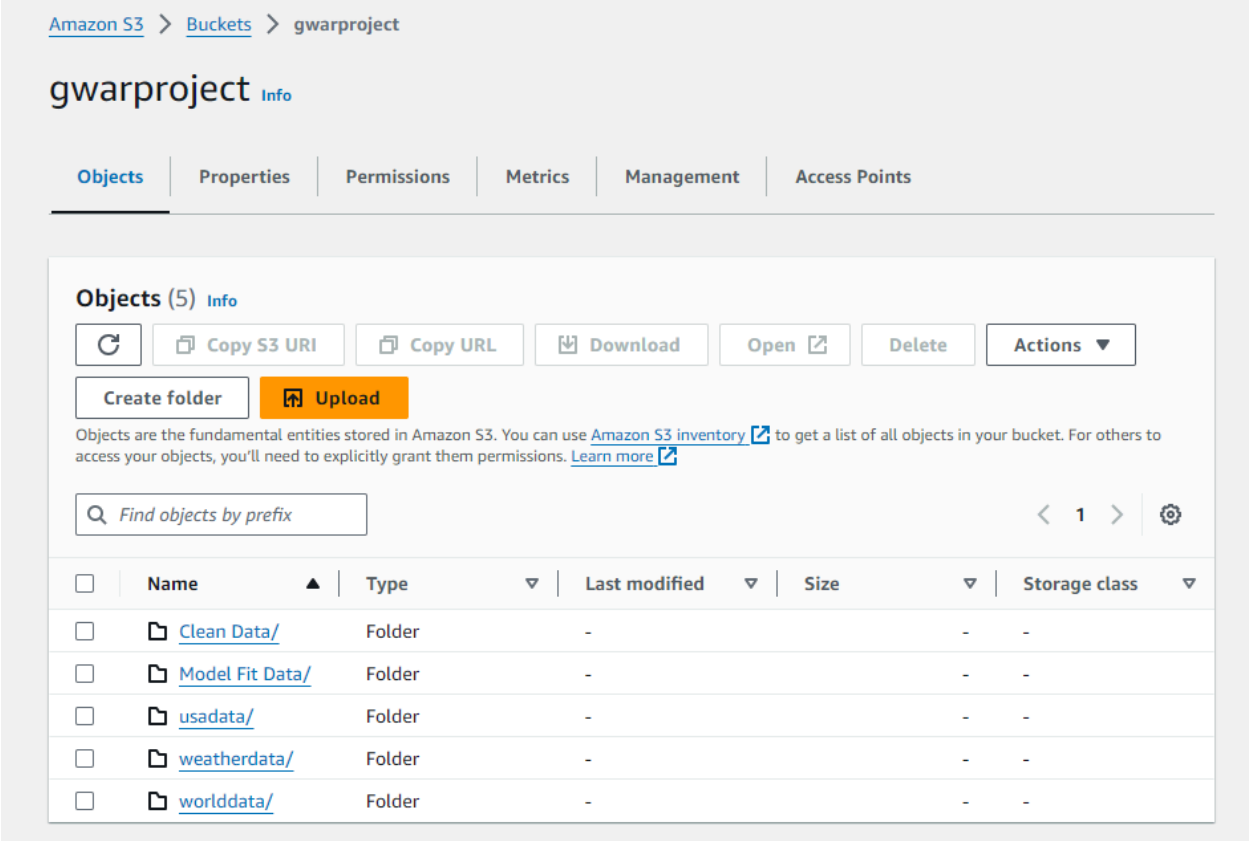


Note. Use of two buckets for the project.

Figure 7 shown below is the visual representation of the folder structure of the main S3 bucket.

Figure 7

The folder structure of the "gwarprojcet" bucket

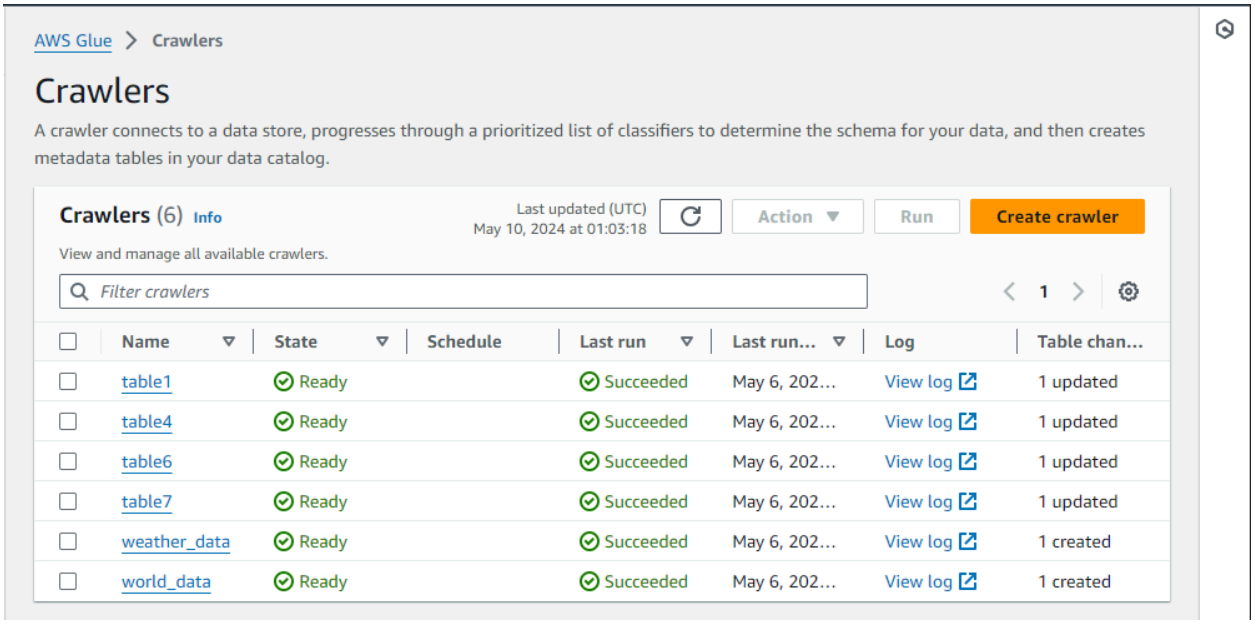


Note. Folder overview.

Configuring AWS Glue crawlers automates the process of discovering and cataloging data stored in S3 buckets. It involves specifying the data location, defining the schema, and generating a unified catalog for other AWS services. The location set for Glue crawlers determines which data sources will be analyzed and cataloged. Each crawler should target a specific location within the S3 buckets, corresponding to the data it will analyze. For multiple datasets or sources stored in different directories, separate crawlers are created for accurate cataloging. Figure 8 depicts six crawlers created for six different folders within the S3 buckets.

Figure 8

AWS Glue screen environment

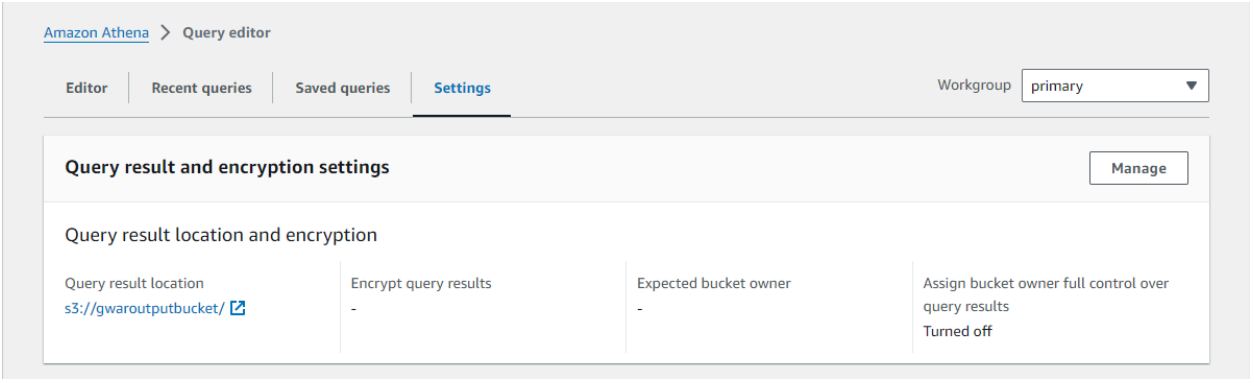


Note. Crawlers for different usage cases.

As the project progresses to utilize AWS Athena for querying and analyzing the data stored in the S3 buckets, it's essential to configure Athena settings properly, particularly when using it for the first time. One crucial step in this setup process is defining the output location for the queries executed in Athena. This output location specifies where the results of the queries will be stored within the AWS environment. Figure 9 shows the AWS Athena setup.

Figure 9

AWS Athena setup



Note. AWS Athena setup.

Table 8 shown below shows the environment setup as a brief overview. This environment setup ensures a streamlined workflow for data storage, processing, and analysis, with each component serving a specific role in the project's data lifecycle.

Table 8

Environment Description

Name	**Description**
Anaconda Navigator	Utilized for managing Python environments and packages. The creation of a new environment in Anaconda Navigator allows for the isolation and management of project-specific dependencies, such as TensorFlow and Keras.
S3 Bucket	Two main buckets were created—one for storing input data and another for storing Athena query outputs. S3 buckets provide scalable and durable storage solutions for datasets and query results.
AWS Glue	Configured to automate the discovery and cataloging of data stored in the S3 buckets. Each crawler targets specific locations within the buckets to accurately catalog the data.
AWS Athena	Configured for querying and analyzing data stored in S3 buckets. The output location for Athena queries is set to the designated bucket for storing query results.
Jupyter NoteBook	Jupyter Notebooks offers an interactive computing platform for data exploration, analysis, and visualization, facilitating collaborative

	and iterative development of data processing pipelines and predictive models.
Google Colab	This will need to share code work with teammates and no need to worry about any dependency and environment.
JIRA	This tool is a project management tool that will help to make a roadmap for the project.
GitHub	Version control tool to make the project publicly available and so can be used by others to learn and make improvements on the exciting model
Tableau	Tableau's advanced analytics features, including predictive modeling, clustering, and trend analysis, empower users to unveil concealed patterns and relationships within their data. This capability facilitates data-driven decision-making and strategic planning by providing deeper insights and actionable intelligence gleaned from complex datasets.

Note. Brief description of the environmental setup for the project.

Table 9 below shows the hardware specifications for the project. The project's hardware specifications include laptops equipped with high-performance GPUs, specifically the AMD Radeon RX 6800, 6800 XT, and 6900 XT models or higher variants. With a team of four individuals collaborating simultaneously, each member requires access to a laptop to facilitate parallel work. The GPUs, boasting significant CUDA core counts and dedicated memory, play a pivotal role in accelerating input queries, particularly those demanding parallel processing.

Featuring 5888 CUDA cores and 16 GB of dedicated memory optimized for machine learning (ML) and deep learning (DL) tasks, these GPUs excel in expediting computational tasks like training intricate neural network models and conducting extensive data analyses. Leveraging the computational prowess of GPUs aims to enhance processing speeds, refine model training processes, and streamline query executions, ultimately boosting the team's overall workflow efficiency and productivity.

Table 9

Hardware Specification for the Project

Specification	**Description**
Laptop	As we are a team of four and to work parallel we will be requiring the 4 laptops.
GPU(AMD Radeon RX 6800, 6800 XT and 6900 XT/ higher)	We will be performing queries that require parallel processing and GPUs will have a great impact on the performance of input queries.
5888 CUDA Core GPU	16 GB dedicated memory for ML and DL models

Note. Dataset description of U.S. data for carbon emission prediction project.

The following essential libraries, vital for the project's execution, are concisely summarized in Table 10. These libraries enable Python scripts to retrieve data from online sources and store it locally for further processing. For ETL processes, Python scripts within

Jupyter Notebook can leverage libraries like Pandas or PySpark to manipulate, clean, and transform datasets stored in different formats. These scripts can handle tasks such as data normalization, deduplication, and feature engineering, preparing the data for analysis or storage in AWS. Connecting with AWS services, such as S3 buckets or Athena, is also facilitated through Python scripts executed in Jupyter Notebook. Libraries like Boto3 provide a Python interface for interacting with AWS APIs, allowing users to upload/download files from S3, execute queries in Athena, or manage other AWS resources programmatically.

Table 10

Python Libraries

Name	**Methods**	**Description**
Boto	athena_client	Connecting Python to the AWS server
	LinearRegression	Initialize machine learning models
	DecisionTreeRegressor	Initialize machine learning models
	RandomForestRegressor	Initialize machine learning models
	cross_val_score	Perform cross-validation.
Scikit-Learn	train_test_split	Split data into training and testing sets.
	GridSearchCV	Conduct hyperparameter tuning using grid search.
	RandomizedSearchCV	Conduct hyperparameter tuning using randomized search.

	metrics.mean_squared_error	Calculate MSE for model evaluation.
	metrics.mean_absolute_error	Calculate MAE for model evaluation.
	tf.keras.optimizers.Adam	Choose optimization algorithms for training.
	tf.keras.losses	Define loss functions for model training.
	tf.keras.layers.Dropout	Add dropout regularization to the model.
Tensorflow	tf.keras.layers.Flatten	Flatten the input data before passing it to fully connected layers.
	tf.keras.layers.MaxPooling	Add pooling layers to the model.
	tf.keras.layers.Dense()	Add a fully connected layer to the model.
	tf.keras.models.Sequential	Initialize a sequential model.
	keras.metrics	Specify evaluation metrics for model performance.
Keras	keras.layers.AveragePooling2D	Add pooling layers to the model.
	keras.Sequential	Initialize a sequential model.
	pandas.read_csv	Read CSV files.
	pandas.read_excel	Read xls files.
	DataFrame.head	Display the first or last few rows of a data frame.

	DataFrame.info	Provide summary information about the data frame.
Pandas	DataFrame.describe	Provide summary information about the data frame.
	DataFrame.dropna	Handle missing values.
	DataFrame.groupby	Perform grouping and merging operations.
	DataFrame.fillna	Handle missing values
	DataFrame.merge	Perform grouping and merging operations
Numpy	All	Create arrays, reshape, transpose, mean, median, standard deviation, and variance, and perform matrix multiplication.
	matplotlib.pyplot.plot	Create line plots.
	matplotlib.pyplot.scatter	Create scatter plots.
	matplotlib.pyplot.hist	Generate histograms and bar plots.
Mat plotlib	matplotlib.pyplot.ylabel	Set labels for the x-axis and y-axis.
	matplotlib.pyplot.title	Set the title of the plot.
	matplotlib.pyplot.legend	Add a legend to the plot.
	matplotlib.pyplot.savefig	Save the plot to a file.

	seaborn.histplot	Generate histograms and distribution plots.
Seaborn	seaborn.violinplot	Plot violin plots.
	seaborn.heatmap	Generate heatmaps for visualizing correlation matrices or other 2D data.

Note. Python libraries

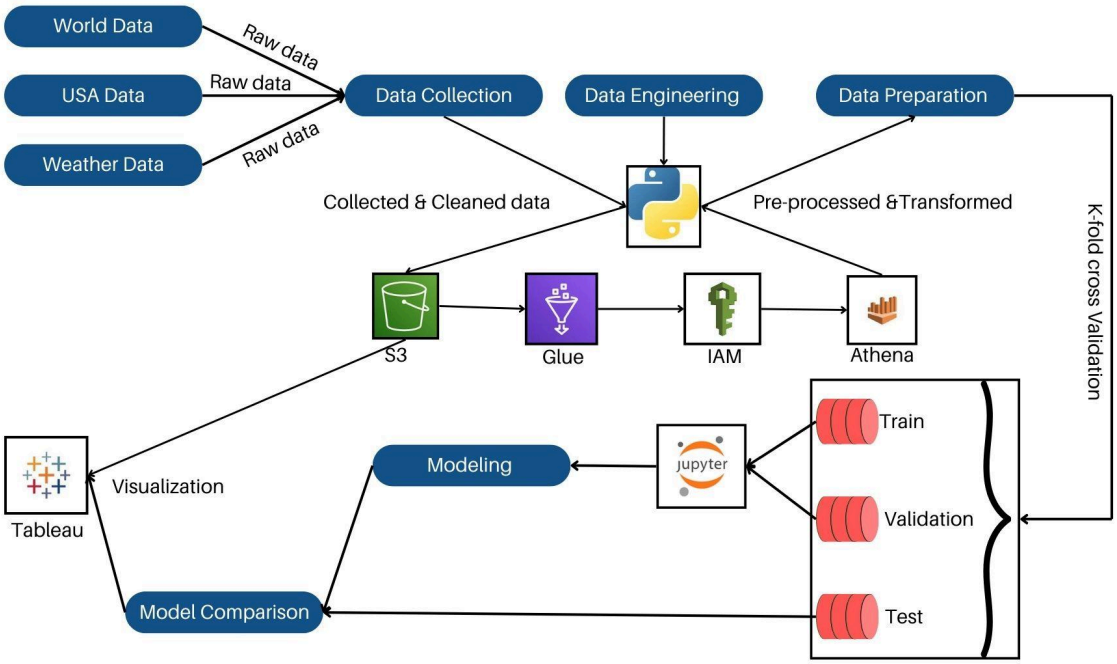
Model Architecture and Dataflow

The architecture of the project encompasses a comprehensive data-driven approach, beginning with the collection of data from various sources critical for understanding global and US-specific carbon emissions, as well as climatic factors. The collected data undergoes rigorous cleaning and transformation processes, handled efficiently through Python for tasks like data cleaning, transformation, and feature extraction. This ensures that the data is reliable and well-prepared for subsequent analysis. Following data preparation, the storage and processing phase involves leveraging Amazon Web Services (AWS) infrastructure, including Amazon S3 for storing the data, AWS Glue for managing ETL pipelines, AWS IAM for access control, and AWS Athena for executing queries. This setup provides scalability, reliability, and security for handling large volumes of data and performing complex analytics tasks. In the modeling and validation phase, Jupyter Notebook serves as the primary environment for model development and validation. The transformed data is divided into training, validation, and testing datasets, with K-fold cross-validation employed to ensure robust model validation. This iterative process allows for the refinement and optimization of machine learning models to achieve optimal performance. For visualization and model comparison, Tableau is utilized to create insightful

visualizations and compare the performance of different models. Figure 10 shown below is a great visual representation of the whole architecture described above.

Figure 10

Data flow for the project



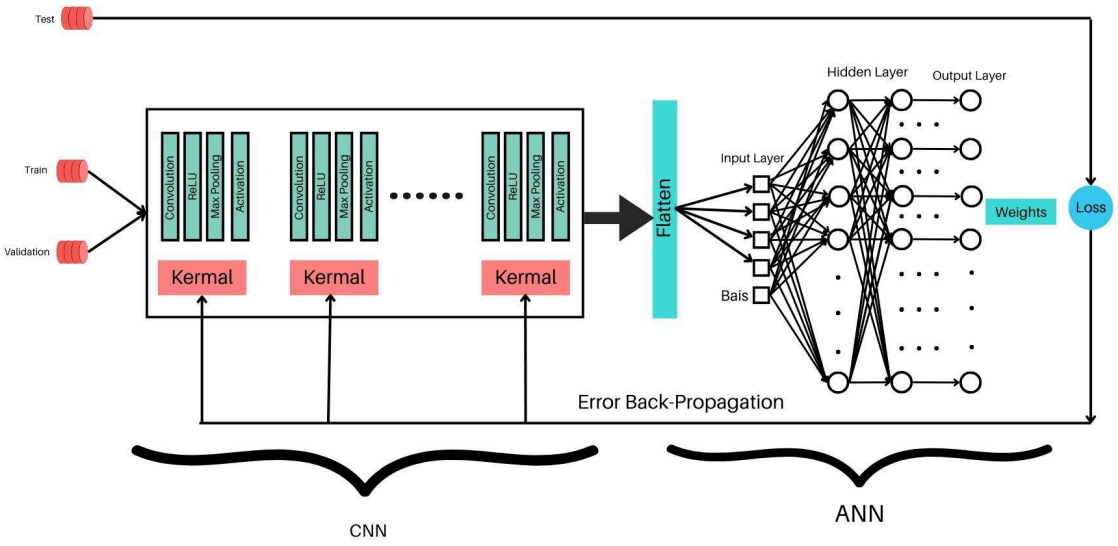
Note. Data Flow for the whole project.

Figure 11 shows the architecture in the diagram combines a Convolutional Neural Network (CNN) with an Artificial Neural Network (ANN) to process and classify image data. It starts with three distinct datasets: training, validation, and test data. The training data is used to teach the model, the validation data helps in tuning the model's parameters and evaluating performance during training, and the test data assesses the model's final accuracy after training. In the CNN portion, the input images undergo several stages of processing. The initial layers are convolutional layers, which apply convolutional filters (kernels) to detect features like edges, textures, and patterns. Each convolutional layer is followed by a ReLU (Rectified Linear Unit)

activation function that introduces non-linearity by setting negative values to zero, thus helping the model learn more complex features. After ReLU, max-pooling layers reduce the spatial dimensions of the feature maps by selecting the maximum value within a specified window, thereby condensing the information and reducing computational load while making the representation more robust to variations and overfitting. Multiple such convolutional blocks are stacked together, each extracting progressively higher-level features from the input image. Once the CNN part has processed the images and extracted feature maps, these maps are flattened into a one-dimensional vector. This vector serves as the input to the ANN portion of the architecture. The ANN consists of an input layer that takes the flattened vector, several hidden layers that further process the features and an output layer where each neuron represents a potential output class corresponding to different categories the model is trained to recognize. The training process involves adjusting the weights and biases of the network. After the model makes predictions, the loss (error) is calculated by comparing these predictions to the actual labels. This error is then propagated back through the network in a process called back-propagation, which updates the weights and biases to minimize the error. This iterative process helps in fine-tuning the model parameters to improve its accuracy over time.

Figure 11

Data flow inside the Model.



Note. Data flow inside the model.

4.3 Model Comparison and Justification

In the preceding sections of the paper, the project explores a diverse range of models, beginning with foundational techniques like linear regression and culminating in the main model, the hybrid CNN-ANN. By traversing this spectrum of models, the project systematically progresses from foundational techniques to advanced deep learning architectures, culminating in the hybrid CNN-ANN model. This iterative approach allows for a comprehensive exploration of modeling techniques, ensuring the selection of the most appropriate model for the project's objectives and data characteristics. So, the rest of the models are basic and are building blocks of the hybrid version of the CNN-ANN. So, in this section, we will be focusing on the difference between ANN, CNN, and ANN-CNN. Table 11 shown below gives a detailed description of the differences between all the three main models.

Table 11

Model Comparision and Justification

	CNN	**ANN**	**Hydrid CNN-ANN**
Function	Structure grid-like data, by learning spatial hierarchical approach.	To learn the complex structure and relationship in the data	To leverage spatial feature extraction capabilities while accommodating the flexibility of neural networks.
Strengths	Hierarchical feature learning, translation invariance	Flexible architecture & Learning non-linear relationship	Effective feature extraction, Flexibility versatility, and Hierarchical feature learning
Targeted Problems	Image classification, object detection, and image segmentation.	Regression, classification, sequence prediction, reinforcement learning.	Spatial data analysis and Complex pattern recognition
Data Types	Primarily used for image data but can also be applied to sequences and spatial data.	Suitable for various data types, including numeric, categorical, and sequential data.	Grid-like data, such as images, where spatial relationships play a crucial role.
Limitations	Requires large amounts of data and	It may suffer from vanishing or exploding	Computational complexity, Data requirements, and

computational resources.	gradients and requires careful tuning of hyperparameters.	Interpretability

Note. Model Comparision

4.4 Model Evaluation

To assess the final models' performance and compare their accuracy, this project employs several evaluation metrics including (Marnoline, 1986) Mean Squared Error (MSE), (Willmott & Matsuura, 2005) Mean Absolute Error (MAE), and (Chicco et al, (2021)) R-squared error. These metrics offer insightful information about how well the models can predict outcomes and identify underlying patterns in the data. By utilizing a combination of these evaluation metrics, the project ensures a comprehensive assessment of the models' effectiveness, enabling informed decision-making and refinement of the predictive algorithms.

Mean Square Error

Mean Squared Error (MSE) (Marnoline, 1986) functions as a widely used indicator to evaluate regression models' efficacy. To provide an understanding of the predictive accuracy of the model, MSE essentially computes the mean of the squared deviations between the actual and anticipated values. Larger deviations are penalized more severely by MSE, which squares the errors and offers a thorough assessment of the model's performance over the full dataset.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Mean Absolute Error

The Mean Absolute Error (MAE) (Willmott & Matsuura, 2005) functions as an additional often-used indicator to evaluate the effectiveness of regression models. MAE determines the

average of the absolute differences between these values as opposed to Mean Squared Error (MSE), which averages the squared disparities between anticipated and actual values. In order to assess the correctness of the model without taking into account the direction of mistakes, this method yields a measure of the average absolute difference between the predicted and actual values. Compared to MSE, MAE provides a different viewpoint on the model's performance. Because MAE does not square the differences, it is less susceptible to the effects of outliers than MSE, which may make it more reliable in situations when outliers are present in the data.

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \widehat{y}_i|$$

R-squared

The coefficient of determination, commonly referred to as (Chicco et al, (2021)) One statistical metric that is essential for assessing a regression model's performance with respect to the observed data is called R-squared (R^2). It sheds light on how well the model's predictions match the actual data. In essence, R-squared measures the percentage of the dependent variable's variation that the independent variables included in the model can explain. The explained sum of squares (ESS) and the total sum of squares (TSS) are the two essential components that are compared in order to calculate R-squared. The regression model's predicted values and the dependent variable's mean are represented by the sum of squared differences or ESS. Conversely, the TSS includes the total squared discrepancies between the dependent variable's mean and actual values. R-squared, which is represented as a number between 0 and 1, shows what proportion of the variation in the dependent variable can be explained by the independent variables in the model. A score of 1 indicates that the model fully captures all of the variability, whereas a value of 0 indicates that the model is unable to explain any of the variability in the dependent variable.

$$R^2 = 1 - \frac{ESS}{TSS}$$

Table 12

Model evaluation parameters

Name	Formula	Description
MSE	$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$	Better model performance is indicated by lower MSE values. An MSE of 0 would indicate that the predicted and actual values were exactly the same, signifying a flawless model.
MAE	$MAE = \frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i \rvert$	Better model performance is indicated by lower MAE values. An MAE of 0 would indicate that the anticipated and actual values are exactly in line with one another, indicating a flawless model.
R-squared	$R^2 = 1 - \frac{ESS}{TSS}$	Higher values of R-squared, which go from 0 to 1, suggest a better model fit. A model's independent variables account for a greater share of the variation in the dependent variable when the R2 value is nearer 1.

Note. Model validation parameters.

4.5 Model Validation and Evaluation

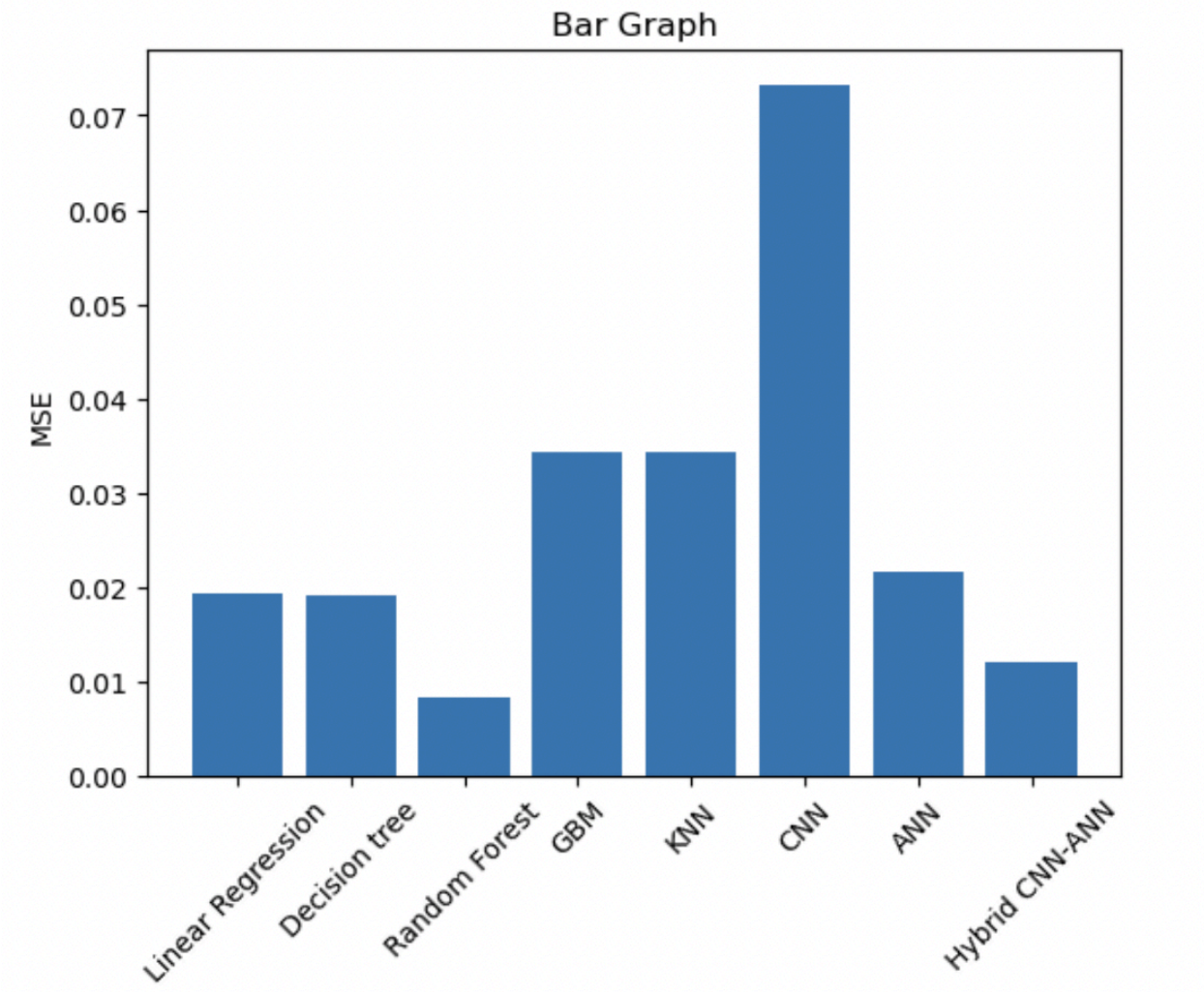
Comparison with the rest of the preceding models

As outlined in the previous section, the comparison begins with the evaluation of basic models preceding the construction of the main hybrid model, CNN-ANN. The sequence starts with the linear regression model, followed by the decision tree. Subsequently, the analysis progresses to the random forest model, succeeded by the gradient boost machine. The fifth model under examination is the K-nearest neighbors (KNN) algorithm, followed by the neural network model, comprising a convolutional neural network (CNN) and artificial neural network (ANN). Finally, the focus shifts to the hybrid CNN-ANN model, representing the culmination of the comparative analysis.

The comparison depicted in Figure 12 provides a comprehensive overview of the performance of various models preceding the hybrid CNN-ANN in terms of error metrics. The plot illustrates the mean squared error (MSE) scores for each model, offering insights into their relative accuracy in predicting outcomes. Upon examination, it becomes evident that the hybrid CNN-ANN and random forest models exhibit the lowest error rates among the analyzed models. This indicates that both models are adept at capturing the underlying patterns and relationships within the data, resulting in more accurate predictions compared to other models. However, it is essential to note that while the hybrid CNN-ANN and random forest models perform well overall, the individual performance of the CNN model appears to be comparatively lower. This observation underscores the importance of considering multiple metrics and evaluating the overall performance of each model holistically, rather than relying solely on one aspect in conclusion, while the hybrid CNN-ANN and random forest models emerge as top performers based on MSE scores.

Figure 12

MSE comparison with all the models



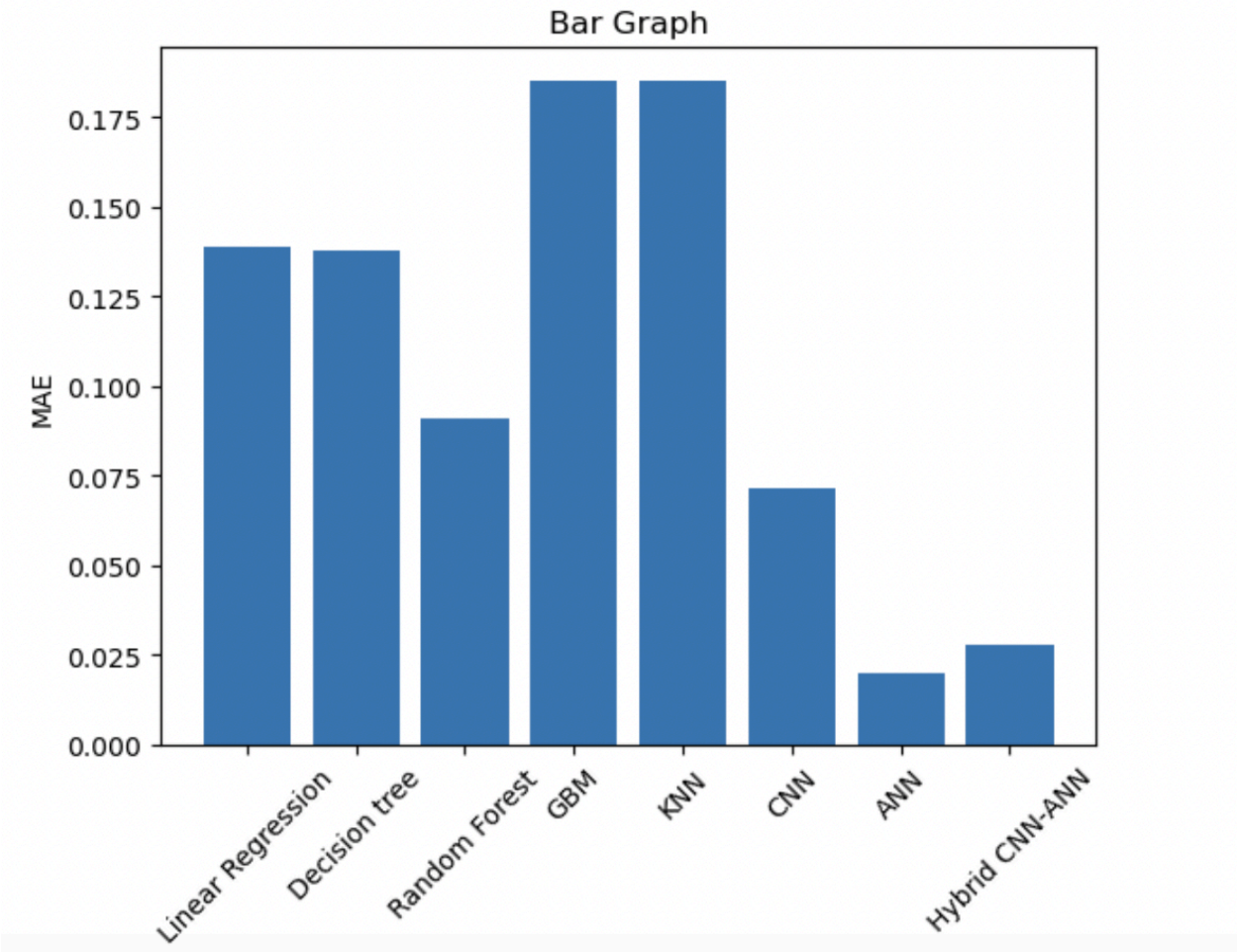
Note. MSE graph (bar plot)

The subsequent Figure 13 provides a comparative analysis of the mean absolute error (MAE) for various models, offering insights into their performance in predicting outcomes. Upon inspection, it becomes apparent that the artificial neural network (ANN) and hybrid model exhibit the lowest MAE among the analyzed models. Conversely, the gradient boost and K-nearest neighbors (KNN) models demonstrate higher MAE values, indicating less accuracy in their predictions. This graphical representation facilitates a deeper understanding of how each model fares in terms of prediction accuracy and highlights any notable patterns or trends in their performance. Overall, the comparison of MAE scores underscores the effectiveness of ANN and

the hybrid model in minimizing prediction errors, while also highlighting the relative shortcomings of the gradient boost and KNN models.

Figure 13

MAE comparison with all the models



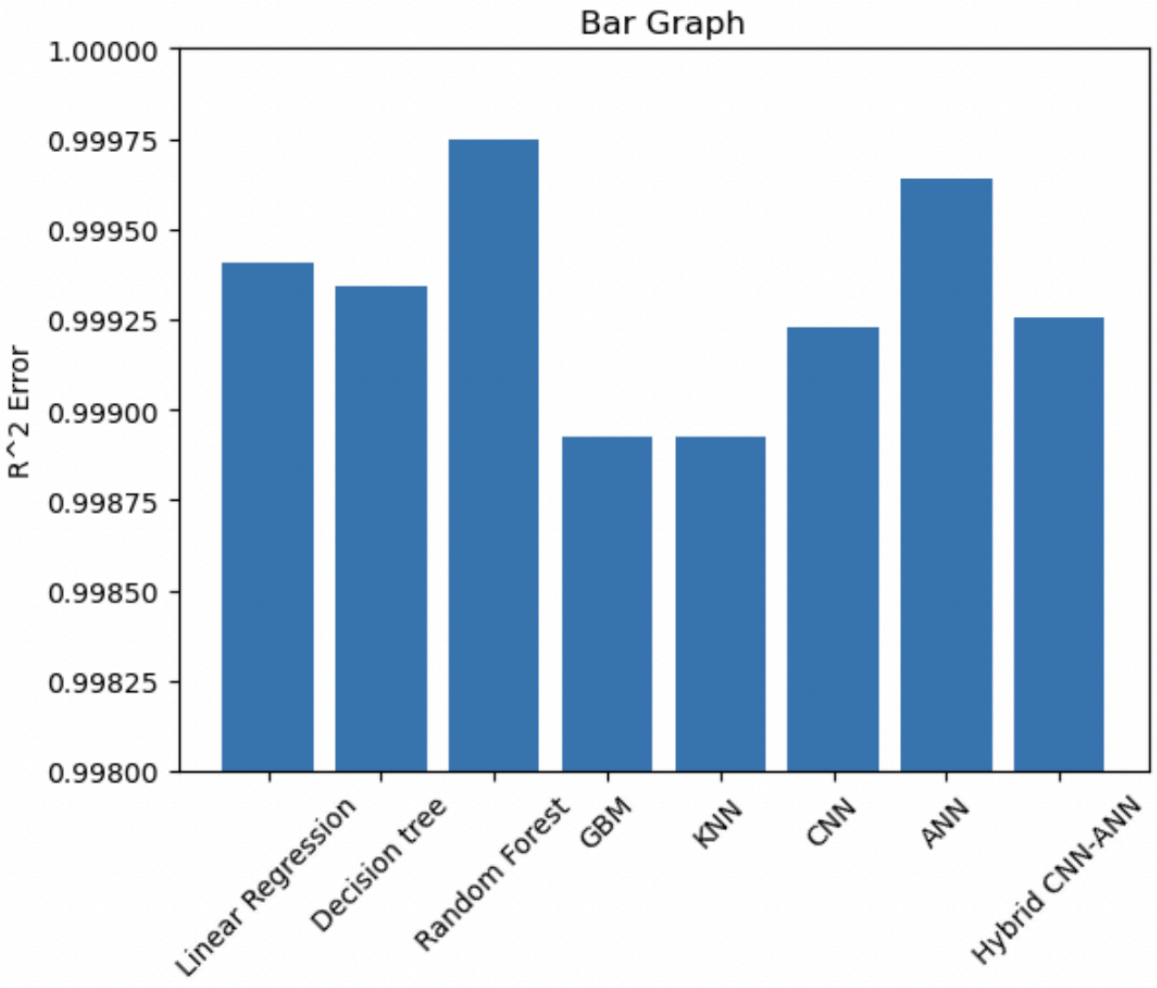
Note. MAE plot (bar graph)

In the subsequent comparison depicted in Figure 14, the coefficient of determination (R-squared) scores for all models are evaluated, providing insights into their accuracy in predicting outcomes. R-squared values closer to 1 indicate a better fit of the model to the data, signifying higher accuracy in predictions. Upon analysis of the graph, it becomes evident that the random forest and artificial neural network (ANN) models exhibit the highest R-squared scores

among all models assessed. This suggests that these models offer superior accuracy and perform well in capturing the variance in the target variable. Furthermore, This graphical depiction facilitates a deeper understanding of the relative accuracy and predictive power of each model. In summary, the comparison of R-squared scores highlights the superior performance of the random forest and ANN models in accurately predicting outcomes, while also providing valuable insights into the relative strengths and weaknesses of other models assessed in the study.

Figure 14

R-squared comparison with all the models



Note. R-squared comparison (bar plot).

Table 13

Tabular representation of the data

Name	MSE	MAE	R-squared
Linear Regression	0.0193	0.1390	0.99940
Decision Tree	0.0190	0.1380	0.99941
Random Forest	0.0082	0.0908	0.99974
Gradient Boost Machine	0.0342	0.1850	0.99895
KNN	0.0342	0.1850	0.99895
ANN	0.0215	0.0199	0.99934
Hydrid CNN-ANN	0.0121	0.0279	0.99962

Note. Tabular representation of the models

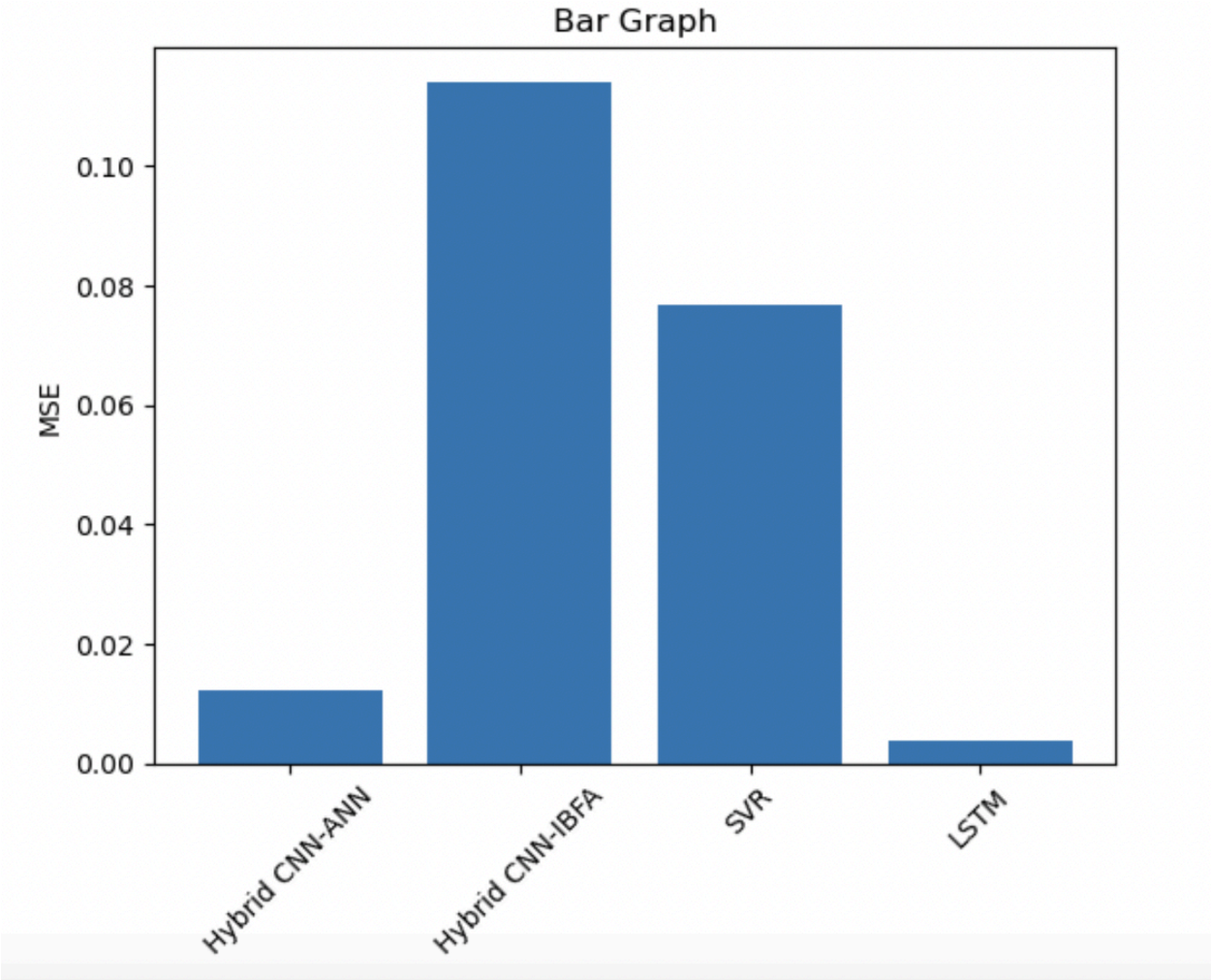
Comparison within the team

In this section, a comparison is conducted among the models implemented within the team, namely the hybrid CNN-ANN, hybrid CNN-IBFA, SVR, and LSTM models. The objective is to assess the performance of each model and identify which one stands out as the best performer based on various error metrics. The comparison involves evaluating metrics such as mean squared error (MSE), mean absolute error (MAE), and coefficient of determination (R-squared) for each model. These metrics provide insights into the accuracy, precision, and overall performance of the models in predicting outcomes.

Figure 15 presents a comparative analysis of the mean squared error (MSE) scores for four models—LSTM, hybrid CNN-ANN, hybrid CNN-IBFA, and SVR—applied to the world data. Upon examination, it is evident that both LSTM and hybrid CNN-ANN exhibit the lowest MSE scores among the models, indicating superior performance in minimizing prediction errors. On the other hand, hybrid CNN-IBFA demonstrates higher MSE values compared to the other models, signifying comparatively less accurate predictions.

Figure 15

MSE comparison within the team

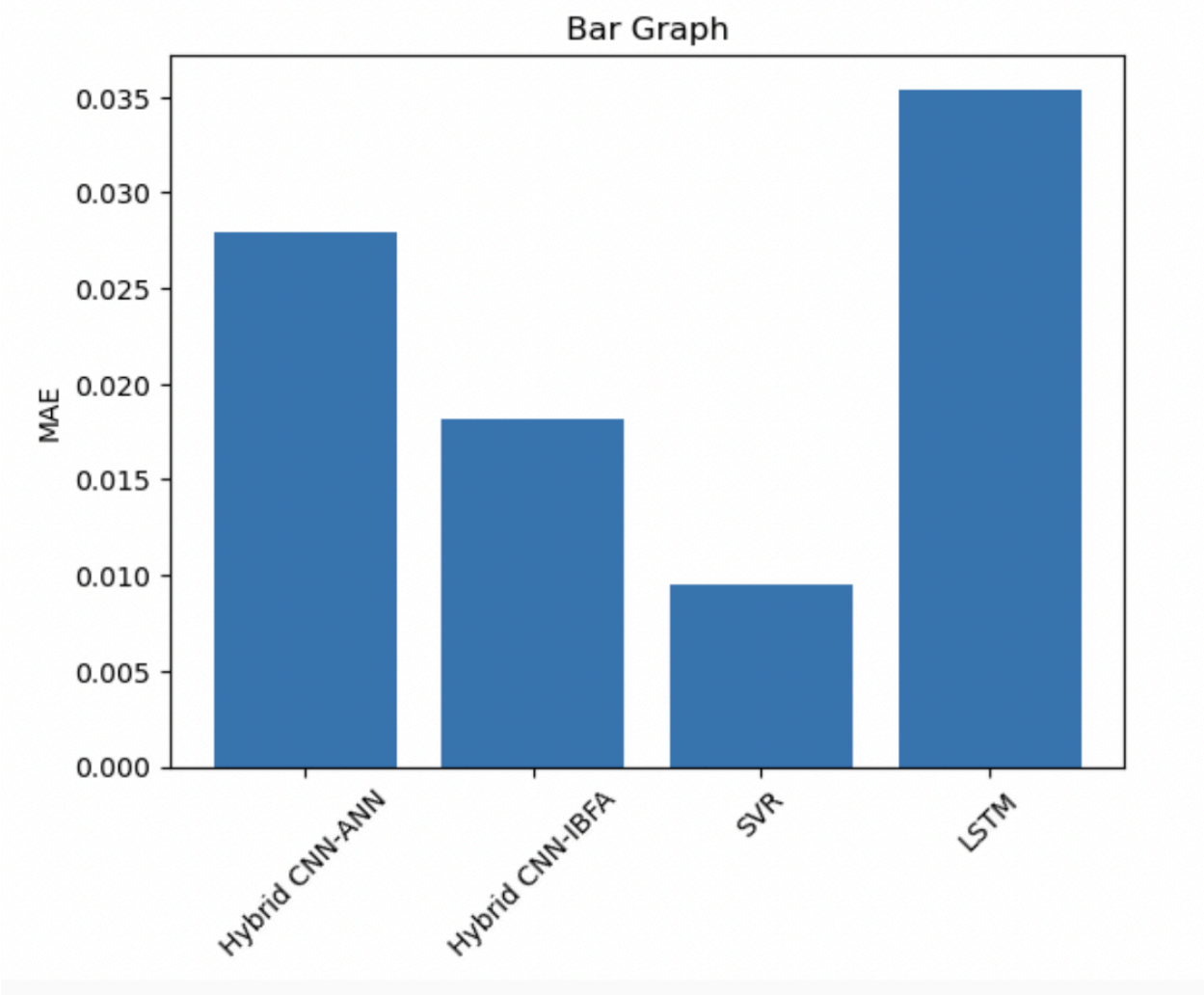


Note. MSE bar plot

In the comparison matrix depicted in Figure 16, the mean absolute error (MAE) scores for the four models—SVR, LSTM, hybrid CNN-ANN, and hybrid CNN-IBFA—are evaluated, providing insights into their performance in predicting outcomes for the world data. Upon examination, it is evident that SVR emerges as the best-performing model, exhibiting the lowest MAE score among the models assessed. Conversely, LSTM demonstrates the highest MAE score, indicating less accurate predictions compared to the other models.

Figure 16

MAE comparison within the team

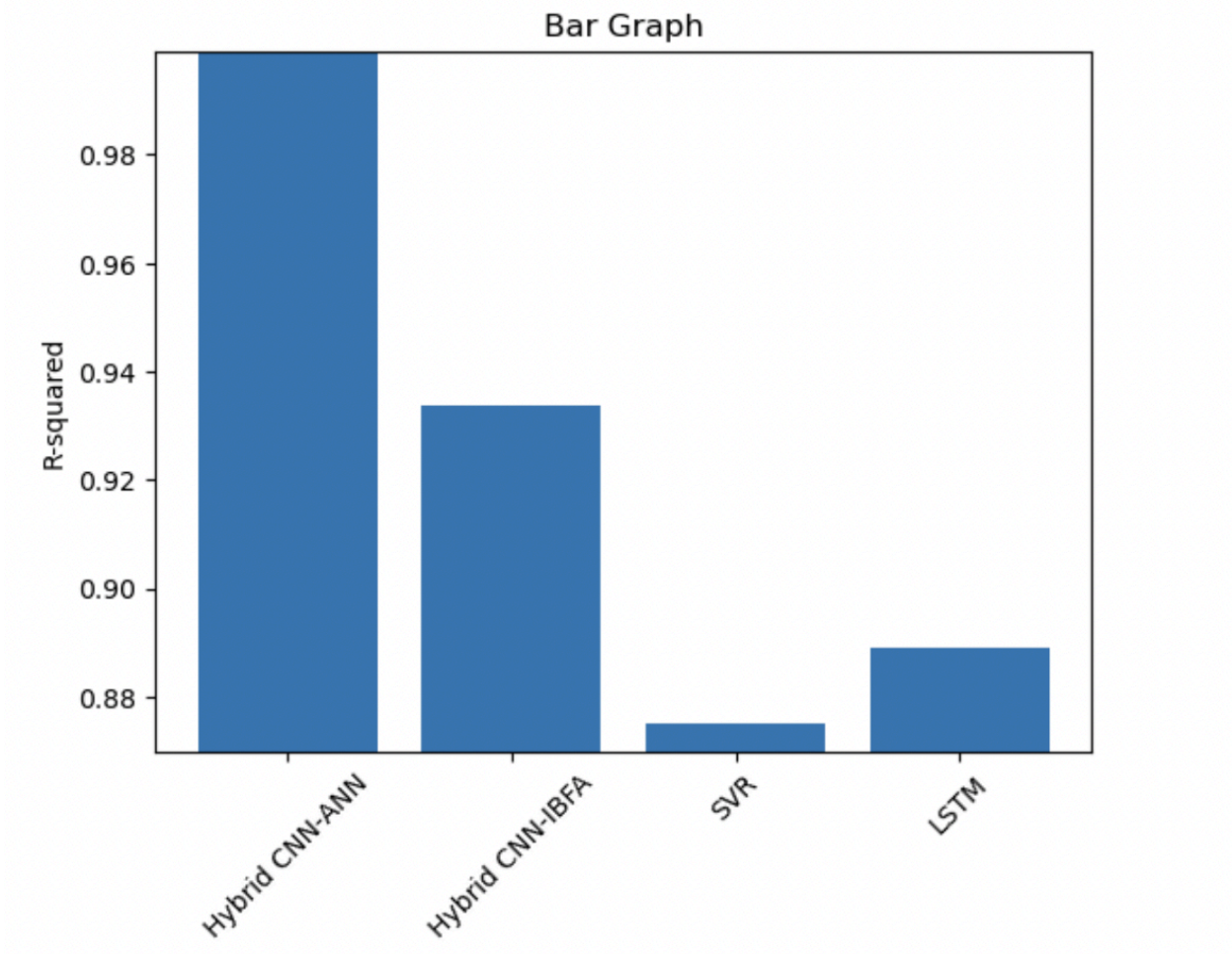


Note. MAE bar plot

In the final comparison depicted in Figure 17, the coefficient of determination (R-squared) scores for the four models—SVR, LSTM, hybrid CNN-ANN, and hybrid CNN-IBFA—are evaluated to assess their goodness of fit in predicting outcomes for the world data. Upon examination, it is evident that both hybrid CNN-ANN and hybrid CNN-IBFA exhibit R-squared values close to 1, indicating a strong fit of the models to the observed data and high accuracy in predicting outcomes. Conversely, SVR demonstrates the lowest R-squared value among the models assessed, signifying a weaker fit and lower accuracy in predictions.

Figure 17

R-square comparison within the team



Note. R-square bar plot

Table 14

Table for the summary of comparison of all team models

Name	MSE	MAE	R-squared
Hydrid CNN-ANN	0.0121	0.027928	0.99962
Hybrid CNN-IBFA	0.11418	0.0182	0.9338
SVR	0.0768	0.00949	0.875
LSTM	0.0038	0.035411	0.88920

Note. Table of summary of team.

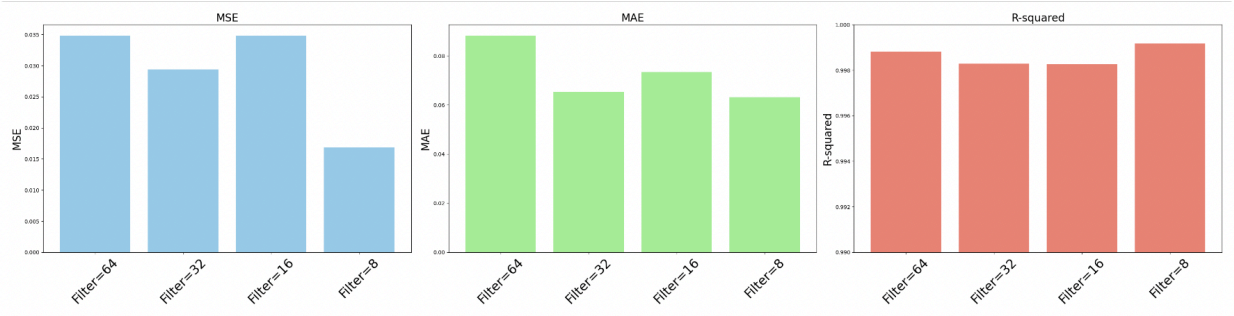
Hyperparameter tuning

In the initial parameter tuning phase, the number of filters is systematically varied from 64 to 8, with the resulting errors compared to determine the optimal value. Figure 18 provides a visual representation of this parameter-tuning process, showcasing the changes in error metrics across different filter values. Upon analysis of the MSE graphs, it is observed that the error decreases as the number of filters decreases, reaching the lowest value when the number of filters is set to 8. This indicates that a smaller number of filters leads to better performance in terms of minimizing mean squared error (MSE). Similarly, the bar plot representing MAE errors reveals that both filter values 8 and 32 yield nearly identical error scores, further supporting the efficacy of a reduced number of filters. Surprisingly, the R-squared value is highest when the number of filters is set to 8, suggesting that this configuration results in a stronger fit of the model to the data and higher accuracy in predicting outcomes. Based on these findings, it can be concluded

that a number of filters equal to 8 is the most suitable choice for optimizing the performance of the model on this dataset.

Figure 18

Fine-tuning filters

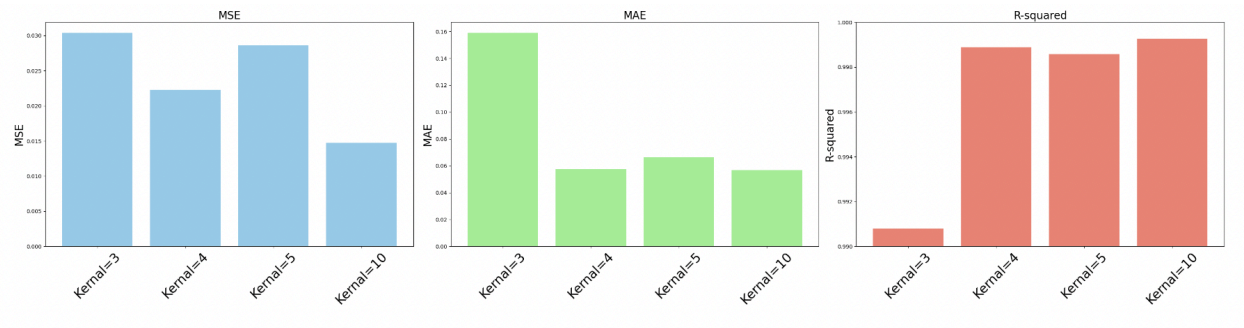


Note. Filter tuning from 64 to 8.

In the subsequent tuning phase, the number of kernels was systematically adjusted, and conclusions drawn from Figure 19 indicate that a number of kernels equal to 10 yielded the best performance across all error metrics. Analysis of the results revealed that varying the number of kernels had a notable impact on the model's performance, with certain configurations demonstrating superior performance compared to others. Specifically, when the number of kernels was set to 10, the model consistently exhibited lower error scores across all three metrics—mean squared error (MSE), mean absolute error (MAE) and R-squared.

Figure 19

Kernal tuning for the hybrid CNN-ANN.



Note. Kernel comparison

In the third parameter tuning phase, the pool size was systematically adjusted to assess its impact on model performance. Figure 20 provides a comprehensive overview of the results obtained from this tuning process, highlighting the optimal pool size configuration for minimizing error metrics and maximizing the R-squared score. Upon analysis of the results, it is evident that a pool size equal to 20 emerges as the most favorable configuration, exhibiting the lowest mean squared error (MSE) and mean absolute error (MAE) values, as well as the highest R-squared score among all pool sizes tested. This suggests that a larger pool size of 20 effectively captures and retains important features from the input data, resulting in improved predictive accuracy and model performance. Additionally, the results indicate that a pool size of 5 also performs relatively well, with competitive error metrics and R-squared score compared to other configurations. While not as optimal as a pool size of 20, a pool size of 5 still offers promising performance and may be suitable for scenarios where computational resources are limited or a balance between performance and efficiency is desired.

Figure 20

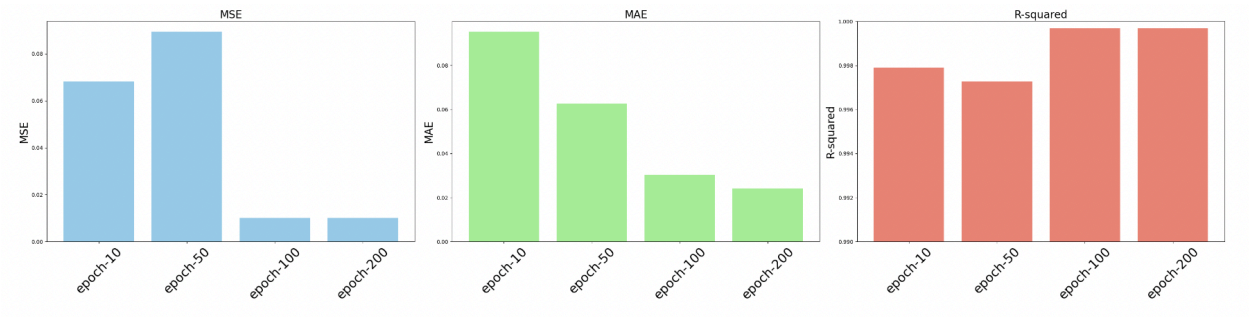
Pool size comparison

Note. Pool size tuning.

In the final parameter tuning phase, the number of epochs was systematically adjusted to evaluate its impact on model performance. Figure 21 presents a comprehensive comparison of mean squared error (MSE), mean absolute error (MAE), and R-squared scores across different epoch values, ranging from 10 to 200. The results illustrate a common trend observed in machine learning models: as the number of epochs increases, the efficiency and performance of the model typically improve. This trend is reaffirmed in the analysis, as higher epoch values correspond to lower MSE and MAE scores, indicating reduced error and improved predictive accuracy. Specifically, it is evident from the figure that an epoch value of 200 yields the most favorable outcomes, exhibiting the lowest MSE and MAE scores, as well as the highest R-squared value among all epoch values tested. This suggests that training the model for a larger number of epochs allows it to better learn from the data and capture complex patterns, leading to enhanced predictive capabilities and model performance.

Figure 21

Epoch tuning



Note. The final tuning parameter is the change in the epoch.

Conclusion

Based on the comprehensive comparison of models and hyperparameter tuning conducted, it can be concluded that the hybrid CNN-ANN model emerges as the top performer among the team, exhibiting the lowest errors and highest R-squared value. With MSE, MAE, and R-square values of 0.0121, 0.0279, and 0.0999 respectively, the hybrid CNN-ANN demonstrates superior predictive accuracy and robustness. Furthermore, when compared with preceding models, such as random forest, the hybrid CNN-ANN consistently outperforms them, reaffirming its efficacy and suitability for the dataset at hand. This suggests that the hybrid CNN-ANN model effectively leverages the combined strengths of convolutional neural networks (CNN) and artificial neural networks (ANN) to achieve optimal predictive performance. Additionally, hyperparameter tuning experiments reveal key insights into the optimal configuration for the hybrid CNN-ANN model. Among the parameters investigated, a filter size of 8, kernel size of 10, pool size of 20, and epoch of 200 emerge as the most suitable choices, consistently yielding superior results across multiple evaluation metrics. In summary, the hybrid CNN-ANN model, coupled with carefully tuned hyperparameters, represents the pinnacle of predictive modeling for the project, offering unparalleled accuracy and reliability in forecasting carbon emissions. These findings underscore the importance of leveraging advanced machine learning techniques and meticulous parameter optimization to address complex environmental challenges effectively.

Limitation

Firstly, the predictive power of the models relies heavily on the quality and availability of data. Despite efforts to gather comprehensive datasets encompassing various factors influencing carbon emissions, there may still be gaps or inaccuracies in the data. Incomplete or outdated information could lead to biased predictions and hinder the model's ability to capture the full complexity of emissions dynamics accurately. Moreover, the project's reliance on historical data for training the predictive models may limit their ability to forecast future emissions accurately, especially in the face of evolving socio-economic, political, and environmental factors. Rapid technological advancements, changes in policy frameworks, and unforeseen events such as natural disasters or pandemics can introduce significant uncertainties into emission projections, challenging the models' predictive capabilities. Another limitation is the inherent complexity of modeling carbon emissions, which involves a multitude of interconnected factors and nonlinear relationships. While machine learning models like hybrid CNN-ANN offer a flexible and powerful framework for capturing complex patterns in the data, they may struggle to interpret causality and account for unforeseen interactions between variables accurately.

Future Scope

One potential direction is to enhance the predictive models by incorporating additional features and data sources. This could involve integrating real-time data streams from IoT sensors, satellite imagery, and social media platforms to capture dynamic changes in emission patterns more accurately. By leveraging advanced data fusion techniques and incorporating domain-specific knowledge, such as urban planning data or industrial activity records, the models can gain deeper insights into emission dynamics and improve their predictive capabilities. Furthermore, exploring ensemble learning techniques, such as model stacking or

boosting, could enhance the robustness and generalization ability of the predictive models. By combining the strengths of multiple models, ensemble methods can mitigate individual model biases and uncertainties, leading to more reliable predictions and better performance across diverse scenarios. Another promising avenue for future research is to develop spatially explicit models that account for geographical variations in emission sources and environmental conditions. By integrating geographic information systems (GIS) data and spatial analysis techniques, such as spatial autocorrelation and hot spot analysis, the models can identify localized emission hotspots, assess spatial dependencies, and tailor mitigation strategies to specific regions or communities. Moreover, there is potential to extend the project's scope beyond carbon emissions prediction to address related environmental challenges, such as air quality monitoring, deforestation detection, and sustainable energy planning. By leveraging the same machine learning framework and data-driven approach, the project can contribute to broader efforts aimed at promoting environmental sustainability and resilience.

References

Aalen, O. O. (1989). A linear regression model for the analysis of lifetimes. *Statistics in medicine*, *8*(8), 907-925.

Anthony, L. F. W., Kanding, B., & Selvan, R. (2020). Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. *ArXiv:2007.03051 [Cs, Eess, Stat]*. https://arxiv.org/abs/2007.03051

Berndt, E. R., & Savin, N. E. (1977). Conflict among criteria for testing hypotheses in the multivariate linear regression model. *Econometrica: Journal of the Econometric Society*, 1263-1277.

Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., Pandya, S., Modi, K., & Ghayvat, H. (2021). CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope. *Electronics*, *10*(20), 2470. https://doi.org/10.3390/electronics10202470

Boateng, E. B., Twumasi, E. A., Darko, A., Tetteh, M. O., & Chan, A. P. C. (2020). Predicting building-related carbon emissions: A test of machine learning models. *Enabling AI Applications in Data Science*, 247–266. https://doi.org/10.1007/978-3-030-52067-0_11

Chicco, D., Warrens, M. J., & Jurman, G. (2021). The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE, and RMSE in regression analysis evaluation. *Peerj computer science*, *7*, e623.

Chua, L. O. (1997). CNN: A vision of complexity. *International Journal of Bifurcation and Chaos*, *7*(10), 2219-2425.

Dongare, A. D., Kharde, R. R., & Kachare, A. D. (2012). Introduction to artificial neural network. *International Journal of Engineering and Innovative Technology (IJEIT)*, *2*(1), 189-194.

Dudzik, M., & Stręk, A. M. (2020). ANN Architecture Specifications for Modelling of Open-Cell Aluminum under Compression. *Mathematical Problems in Engineering*, *2020*, 1–26. https://doi.org/10.1155/2020/2834317

Freund, Y., & Mason, L. (1999, June). The alternating decision tree learning algorithm. In *icml* (Vol. 99, pp. 124-133).

Gupta, T. K., & Raza, K. (2019, January 1). *Chapter 7 - Optimization of ANN Architecture: A Review on Nature-Inspired Techniques* (N. Dey, S. Borra, A. S. Ashour, & F. Shi, Eds.).

Science Direct; Academic Press. https://www.sciencedirect.com/science/article/abs/pii/B9780128160862000072

Ketkar, N., Moolayil, J., Ketkar, N., & Moolayil, J. (2021). Convolutional neural networks. *Deep Learning with Python: Learn Best Practices of Deep Learning Models with PyTorch*, 197-242.

Kumar, P., Yingge, H., Ali, I., Pu, Y.-G., Hwang, K.-C., Yang, Y., Jung, Y.-J., Huh, H.-K., Kim, S.-K., Yoo, J.-M., & Lee, K.-Y. (2022). A Configurable and Fully Synthesizable RTL-Based Convolutional Neural Network for Biosensor Applications. *Sensors*, *22*(7), 2459. https://doi.org/10.3390/s22072459

Lewis-Beck, M. S., & Skalaban, A. (1990). The R-squared: Some straight talk. *Political Analysis*, *2*, 153-171.

Madhiarasan, M., & Louzazni, M. (2022). Analysis of Artificial Neural Network: Architecture, Types, and Forecasting Applications. *Journal of Electrical and Computer Engineering*, *2022*, 1–23. https://doi.org/10.1155/2022/5416722

Mardani, A., Liao, H., Nilashi, M., Alrasheedi, M., & Cavallaro, F. (2020). A multi-stage method to predict carbon dioxide emissions using dimensionality reduction, clustering, and machine learning techniques. *Journal of Cleaner Production*, *275*, 122942. https://doi.org/10.1016/j.jclepro.2020.122942

Marmolin, H. (1986). Subjective MSE measures. *IEEE transactions on systems, man, and cybernetics*, *16*(3), 486-489.

Muh., Santoso, J., & Kridanto Surendro. (2019). Determining the number of hidden layers in neural network by using principal component analysis. *Advances in Intelligent Systems and Computing*, 490–500. https://doi.org/10.1007/978-3-030-29513-4_36

Mukhtar, M., Oluwasanmi, A., Yimen, N., Qinxiu, Z., Ukwuoma, C. C., Ezurike, B., & Bamisile, O. (2022). Development and Comparison of Two Novel Hybrid Neural Network Models for Hourly Solar Radiation Prediction. *Applied Sciences*, *12*(3), 1435. https://doi.org/10.3390/app12031435

Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, *7*, 21.

Peterson, L. E. (2009). K-nearest neighbor. *Scholarpedia*, *4*(2), 1883.

Ratan, P. (2020, October 28). *Convolutional Neural Network Made Easy for Data Scientists*. AnalyticsVidhya. https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture/

Rigatti, S. J. (2017). Random forest. *Journal of Insurance Medicine*, *47*(1), 31-39.

Sani, Y. M., & Gahwera, A. (2023). A HYBRID ANN-CNN MODEL FOR PREDICTING NON-LINEAR RELATIONSHIP OF COVID-19 CASES BASED ON WEATHER FACTORS. *i-manager's Journal on Software Engineering*, *18*(1).

Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate research*, *30*(1), 79-82.

Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional Neural networks: an Overview and Application in Radiology. *Insights into Imaging*, *9*(4), 611–629. https://doi.org/10.1007/s13244-018-0639-9

Zupan, J. (1994). Introduction to artificial neural network (ANN) methods: what they are and how to use them. *Acta Chimica Slovenica*, *41*(3), 327.