MLD Advance Project Phase 3

Topic: Weather Classification using ML

By: 1. Vishwashree V Karhadkar [Student ID: 202307962]

2. Gagandeep Singh [Student ID: 202303876]

INTRODUCTION

In this phase 3 of our project, we classify weather conditions using a multiclass classification approach on a selected dataset[1]. We have chosen to aim for option 2 which includes a pre-trained lightweight deep learning model (RESNET50)[4] and a pipeline tool developed to pretrain the model named X-visionHelper[3]. For this, we will be splitting our main weather dataset into 3 subsets: **A1**, **A2**, and **B**. The dataset will be divided randomly and the split looks like 70% for **A1**, 10% for **A2**, and 20% for **B**. **A1** will be used for the training set **A2 will** be the validation set for tuning and the B set will be the test set.

The primary goal of this report is to present results for our hypothesis of the standard task of classifying weather with option 2, with this, the following report also includes some description information about the dataset[1], various tools and technologies used, and includes results with statistical analysis. This includes an exploration of the data distributions, model evaluation metrics, and the overall accuracy of the classification task. The statistical analysis will involve evaluating the model's predictions using performance metrics such as accuracy, precision, and recall. The Results at the end display how various prediction models with their selected features perform on the given dataset with all metrics listed such as accuracy, etc. which give us insight into how well we are progressing and how the models are performing which can be analyzed for the dataset.

1. Materials and Methods

1.1.1 Dataset Description:

We have fetched a weather-related dataset consisting of weather images into various folders. The folders consist of a few values like dew, rain, rainbow frost, lighting, snow, etc. It is a compatible dataset for our scope of project with the classification task and for supervised learning. All the images are sorted in a particular order, so each folder has a particular weather condition.

The dataset's main source where it was assembled comes from the **Harvard dataset Archives**[1]. It was published on **Nov 10, 2021[1].** It was established for the Earth and Environmental Sciences Department use case [1]. A research article published using the same dataset showed a classification accuracy of **92.68%** using the **MeteCNN** model[5].

Exploring the context of the dataset in more granular way we can express it as follows; the Dataset consists of a total of **6,862** images of various weather conditions collectively added in **11** different folders which are named according to weather conditions and have those specific weather images in them[1]. All images in this dataset are in **JPG** format type; All images are in **colored** (RGB) format and the size of images ranges from <u>3-5 kb to 3-5 MB</u>.[1] The total size of the dataset is **636.73 MB**.[1] These provide a diverse set of visual data to analyze helping us to train our ML models and help predict weather which is the primary objective of our advanced project.

The Listed weather conditions folder names with the count of images within it are as follows:

Dew: 698 files	Rain: 526 files
Fog/Smog: 851 files	Rainbow: 232 files
Frost: 475 files	Rime: 1,160 files
Glaze: 639 files	Sandstorm: 692 files
Hail: 591 files	Snow: 621 files
Lightning: 377 files	

1.1.2 Target Variable Description:

The variables of interest for this project's dataset are a list of folders elaborating weather conditions that we are predicting and classifying as "Weather Condition classification", which is the type of weather depicted in the images of the dataset[1]. The target variable is a **MULTICLASS CLASSIFICATION**, the main objective of which is to predict one of several predefined weather categories based on the input features given in folders of the dataset (primarily image data).

1.1.3 Dataset Splits:

In this project phase 3, the dataset was divided into three subsets: **A1**, **A2**, and **B**, to optimize the model's training, validation, and testing phases. The **A1** subset, which was used as the training set for the deep learning model, consisted of approximately **60%** of the total dataset, or 4,122 images. The **A2** subset, which served as the validation set, contained **10%** of the dataset, or 687 images, and played an important role in tuning the model's hyperparameters during training. Finally, the **B** subset, reserved for testing, consisted of **30%** of the dataset, or 2061 images, and allowed for an unbiased evaluation of the model's performance on unseen data. These percentages are based on an initial random split and can be adjusted later according to the dataset size and project needs, ensuring a balanced approach to training, validation, and testing.

1.2 Machine Learning

For this advanced project's phase 3, we selected option 2, which utilized 3 different software suitable for analyzing and classifying weather conditions based on the provided dataset. Specifically, we employed **DF-Analyze**[2] and **X-visionHelper**[3], which uses **Resnet50**[4] to serve various roles in our machine-learning workflow and execution.

The df-analyze is a Python library package that offers different and multiple utility functions and switches for processing and analyzing datasets for machine learning and prediction. It is a command line tool and can perform ML algorithms on different datasets which are small to medium-sized tabular datasets (less than 200000 samples and from 50-10 features. Df-analyze automates and runs different ML algorithms and gives extensive and elaborate reports. It includes many important key functionalities which are feature type inference, feature description (e.g. univariate associations and stats), data cleaning (e.g. NaN handling and imputation), training, validation, and test splitting, feature selection, hyperparameter tuning, model selection, and validation. It runs the ML algorithms to proceed with an output with all important results stored in tabular format.[2]

X-VisionHelper is a utility designed to fine-tune the ResNet-50[4] model on custom datasets, facilitating the extraction of embeddings for image classification tasks[3]. According to the information stated on the readme, X-VisionHelper gives us the convenience of fine-tuning ResNet-50[4] on different datasets directly without having to write or integrate every dependent and required component from scratch for our ease to get right into tuning our model[3]. This tool was instrumental in adapting the ResNet-50[4] model to our specific dataset, allowing for the extraction of meaningful features for multiclass classification.

ResNet-50[4] which is used in X-VisionHelper[3] has been introduced by the Microsoft research team as a deep neural network which is designed for image recognition using deep residual learning[4]. It is used for addressing vanishing gradients using residual blocks with shortcut connections. The architecture which includes 50 layers makes it highly effective for image classification tasks[4]. The specific Resnet 50 pre-trained models are used on a large scale for transfer learning which offers a good generalization and efficiency[4]; which makes a wise approach to choosing option 2 which includes the above Resnet 50 technology.

By integrating these tools, we established a multiclass classification pipeline that efficiently handled feature extraction, embedding, and analysis, leading to the multiclass classification of weather conditions in our dataset.

In our previous phase, we hypothesized that we would be using ML techniques to classify weather conditions using df-analyze[2] based on the dataset. After dividing the main dataset files into a training set(A1), validation set(A2), and testing set(B) using Python, with the first, specific Run on X-VisionHelper[3], we extracted the embeddings and files and inputted the CSV files to df-analzye[2] to get the results. Based on this, we received the results and found interesting findings and compared them with the results from the previous phase which are discussed thoroughly below in this report.

1.3 Statistical Analysis

The statistical analysis that is used in this project involves using df-analyze[2] machine learning models which have various feature selection techniques that show predictive outcomes. Various models such as Logistic Regression (LR), Stochastic Gradient Descent (SGD), Random Forest (RF), LightGBM (LGBM), and K-Nearest Neighbors (KNN) were used on the datasets. Each model was applied with various feature selection strategies, which include association-based (assoc), predictive (pred), embedded methods (embed_linear, embed_lgbm), and wrapper-based (wrap) models.

For this phase of the project, we will be moving forward and focusing on the **overall accuracy** of the predicted features. It is found in the results directory, a table named 5-fold performance on the holdout set in the file named “results_report.md”.

The statistical analysis involves using the X-VisionHelper[3] which is a pre-trained light-weight deep learner model, that helps in the modelling and testing of the dataset provided. XVisionHelper was used on the dataset itself to get the model weights from the first run and another to get the embeddings and 2 CSV files. Then, we used these embeddings and CSV files to run the df-analyze. We have focused on the total accuracy from the final results given by the df-analyze and found that the model has predicted the accuracy less as compared to the previous phase. K-fold validation was performed using df-analzye and the maximum accuracy that we can see from the table is **49.7%**, shown by lgbm with embed_linear. The results are given below in **Table 1.3.1.**

Table: 1.3.1 5-Fold Performance on holdout set			
Model	**Selection**	**Embed Selector**	**Acc**
lgbm	embed_linear	linear	0.497
lgbm	none	none	0.496
lgbm	embed_lgbm	lgbm	0.496
lgbm	assoc	none	0.494
rf	embed_lgbm	lgbm	0.469
rf	none	none	0.466
rf	embed_linear	linear	0.457
rf	assoc	none	0.456
lgbm	pred	none	0.451
lgbm	wrap	none	0.419

rf	wrap	none	0.402
rf	pred	none	0.382
knn	pred	none	0.360
knn	embed_linear	linear	0.351
knn	none	none	0.351
knn	assoc	none	0.351
knn	embed_lgbm	lgbm	0.351
knn	wrap	none	0.343
sgd	wrap	none	0.285
sgd	none	none	0.230
sgd	embed_linear	linear	0.227

sgd	pred	none	0.226
lr	pred	none	0.217
lr	embed_linear	linear	0.217
lr	none	none	0.217
lr	assoc	none	0.217
lr	wrap	none	0.217
lr	embed_lgbm	lgbm	0.216
sgd	assoc	none	0.203
dummy	assoc	none	0.184
dummy	embed_lgbm	lgbm	0.184
dummy	pred	none	0.184

dummy	none	none	0.184
dummy	embed_linear	linear	0.184
dummy	wrap	none	0.184
gandalf	embed_lgbm	lgbm	0.174
sgd	embed_lgbm	lgbm	0.174
gandalf	none	none	0.109
gandalf	assoc	none	0.091
gandalf	embed_linear	linear	0.077
gandalf	wrap	none	0.063
gandalf	pred	none	0.055

2. Results

2.1 Summary results with Table.

In this phase, in terms of the accuracy of deep learning of option 2 in our case, we have got 44.35%. As this method doesn't save any results in any textual format it is only shown in the image format given below.

```
(xVisionHelper) gagandeepsingh@Gagandeeps-MacBook-Air X-vision-helper % python3 model_finetuning.py --num_classes 11 --num_epochs 2 --batch_size 15 --learning_rat
0.001 --train_dir /Users/gagandeepsingh/Documents/MLD/dataset_sort1/train --val_dir /Users/gagandeepsingh/Documents/MLD/dataset_sort1/val --test_dir /Users/gagand
psingh/Documents/MLD/dataset_sort1/test
/Users/gagandeepsingh/Documents/MLD/xVisionHelper/lib/python3.10/site-packages/torchvision/models/_utils.py:135: UserWarning: Using 'weights' as positional parame
r(s) is deprecated since 0.13 and may be removed in the future. Please use keyword parameter(s) instead.
  warnings.warn(
100%|                                                                     | 253/253 [11:59<00:00,  2.85s/it, loss=2.3
100%|                                                                            | 127/127 [02:04<00:00,  1.02it/
INFO | [EPOCH 1] Training Loss= 2.3099275737883076 Validation Loss=6.8560149896712534 | Training Accuracy=0.22503307461738586 val=0.820634663105011
100%|                                                                     | 253/253 [10:23<00:00,  2.46s/it, loss=2.0
100%|                                                                            | 127/127 [01:56<00:00,  1.09it/
INFO | [EPOCH 2] Training Loss= 2.1874727312284024 Validation Loss=6.319250296978724 | Training Accuracy=0.3488803803920746 val=1.341269612312317
INFO | Model Saved to model_weights.pth
100%|                                                                            | 127/127 [01:56<00:00,  1.09it/
INFO | Test accuracy: 44.35694885253906%
(xVisionHelper) gagandeepsingh@Gagandeeps-MacBook-Air X-vision-helper %
```

The following table is the output after running df-analyze[2] which is summarized below. **Table 2.1.1** shows the performance on holdout performance and **Table 2.1.2** shows the 5-fold performance on the holdout set.

Each model was evaluated based on several metrics, including Accuracy (Acc), Area Under the Receiver Operating Characteristic Curve (AUROC), Balanced Accuracy (Bal-Acc), F1 Score (F1), Negative Predictive Value (NPV), Positive Predictive Value (PPV), Sensitivity (Sens), and Specificity (Spec). Overall, several models demonstrated low performance on our dataset ranging from 5-49% accuracy.

The last table displayed in section no as **Table 2.2.3** showcases the accuracy results from phase 2 previously completed on the same given dataset.

Table: 2.1.1 Holdout set performance										
model	selection	embed_selector	acc	auroc	bal-acc	f1	npv	ppv	sens	spec
lgbm	embed_lgbm	lgbm	0.531	0.833	0.440	0.441	0.952	0.449	0.440	0.952
lgbm	embed_linear	linear	0.524	0.833	0.429	0.431	0.952	0.444	0.429	0.951
lgbm	assoc	none	0.521	0.832	0.428	0.431	0.951	0.491	0.428	0.951
lgbm	none	none	0.519	0.830	0.425	0.426	0.951	0.438	0.425	0.951
rf	embed_lgbm	lgbm	0.489	0.803	0.407	0.404	0.948	0.404	0.407	0.948
rf	none	none	0.487	0.806	0.407	0.401	0.948	0.401	0.407	0.948
rf	embed_linear	linear	0.484	0.799	0.404	0.398	0.948	0.398	0.404	0.947
rf	assoc	none	0.481	0.800	0.406	0.403	0.947	0.406	0.406	0.947
lgbm	pred	none	0.465	0.793	0.370	0.374	0.946	0.394	0.370	0.945

rf	wrap	none	0.420	0.764	0.344	0.341	0.941	0.342	0.344	0.941
lgbm	wrap	none	0.418	0.775	0.331	0.330	0.941	0.339	0.331	0.940
knn	embed_linear	linear	0.381	0.695	0.296	0.296	0.937	0.313	0.296	0.936
knn	none	none	0.381	0.695	0.296	0.296	0.937	0.313	0.296	0.936
knn	assoc	none	0.381	0.695	0.296	0.296	0.937	0.313	0.296	0.936
rf	pred	none	0.380	0.732	0.294	0.294	0.937	0.306	0.294	0.936
knn	pred	none	0.378	0.697	0.292	0.292	0.937	0.307	0.292	0.936
knn	wrap	none	0.374	0.698	0.289	0.288	0.936	0.307	0.289	0.936
knn	embed_lgbm	lgbm	0.365	0.690	0.282	0.282	0.935	0.298	0.282	0.935
sgd	wrap	none	0.282	0.673	0.200	0.187	0.927	0.228	0.200	0.925
lr	embed_linear	linear	0.219	0.550	0.121	0.070	0.927	0.318	0.121	0.913

lr	pred	none	0.219	0.550	0.121	0.070	0.927	0.319	0.121	0.913
lr	assoc	none	0.219	0.550	0.121	0.070	0.927	0.318	0.121	0.913
lr	none	none	0.219	0.550	0.121	0.070	0.927	0.318	0.121	0.913
sgd	embed_ linear	linear	0.218	0.543	0.156	0.134	0.920	0.232	0.156	0.920
lr	wrap	none	0.217	0.551	0.119	0.065	0.928	0.369	0.119	0.913
lr	embed_ lgbm	lgbm	0.217	0.528	0.118	0.071	0.927	0.354	0.118	0.913
gandalf	pred	none	0.189	0.543	0.098	0.044	0.925	0.197	0.098	0.911
dummy	assoc	none	0.184	0.500	0.091	0.028	0.918	0.184	0.091	0.909
dummy	embed_ lgbm	lgbm	0.184	0.500	0.091	0.028	0.918	0.184	0.091	0.909
dummy	embed_ linear	linear	0.184	0.500	0.091	0.028	0.918	0.184	0.091	0.909
dummy	wrap	none	0.184	0.500	0.091	0.028	0.918	0.184	0.091	0.909

dummy	pred	none	0.184	0.500	0.091	0.028	0.918	0.184	0.091	0.909
dummy	none	none	0.184	0.500	0.091	0.028	0.918	0.184	0.091	0.909
sgd	none	none	0.178	0.570	0.159	0.146	0.917	0.210	0.159	0.917
sgd	assoc	none	0.140	0.531	0.133	0.121	0.912	0.195	0.133	0.912
sgd	pred	none	0.140	0.562	0.117	0.107	0.912	0.236	0.117	0.913
sgd	embed_ lgbm	lgbm	0.124	0.506	0.102	0.095	0.909	0.166	0.102	0.909
gandalf	embed_ linear	linear	0.120	0.551	0.112	0.048	0.913	0.234	0.112	0.912
gandalf	embed_ lgbm	lgbm	0.117	0.605	0.117	0.063	0.913	0.144	0.117	0.912
gandalf	assoc	none	0.091	0.522	0.075	0.027	0.869	0.076	0.075	0.907
gandalf	wrap	none	0.076	0.565	0.091	0.018	0.911	0.034	0.091	0.909
gandalf	none	none	0.046	0.631	0.120	0.042	0.912	0.174	0.120	0.911

Table: 2.2.2. 5-fold performance on holdout set										
Model	**Selection**	**Embed Selector**	**Acc**	**AUROC**	**Bal-Acc**	**F1**	**NPV**	**PPV**	**Sens**	**Spec**
lgbm	embed_linear	linear	0.497	0.837	0.405	0.406	0.949	0.459	0.405	0.948
lgbm	none	none	0.496	0.836	0.404	0.405	0.949	0.468	0.404	0.948
lgbm	embed_lgbm	lgbm	0.496	0.840	0.407	0.410	0.949	0.461	0.407	0.948
lgbm	assoc	none	0.494	0.838	0.401	0.401	0.949	0.454	0.401	0.948
rf	embed_lgbm	lgbm	0.469	0.799	0.396	0.393	0.946	0.399	0.396	0.946
rf	none	none	0.466	0.805	0.386	0.382	0.946	0.393	0.386	0.945
rf	embed_linear	linear	0.457	0.802	0.381	0.376	0.945	0.386	0.381	0.945
rf	assoc	none	0.456	0.798	0.388	0.385	0.945	0.396	0.388	0.945
lgbm	pred	none	0.451	0.801	0.355	0.355	0.944	0.409	0.355	0.943
lgbm	wrap	none	0.419	0.765	0.328	0.325	0.941	0.376	0.328	0.940

rf	wrap	none	0.402	0.748	0.321	0.319	0.939	0.327	0.321	0.939
rf	pred	none	0.382	0.742	0.298	0.298	0.937	0.313	0.298	0.936
knn	pred	none	0.360	0.700	0.280	0.278	0.935	0.324	0.280	0.934
knn	embed_linear	linear	0.351	0.699	0.273	0.271	0.934	0.314	0.273	0.933
knn	none	none	0.351	0.699	0.273	0.271	0.934	0.314	0.273	0.933
knn	assoc	none	0.351	0.699	0.273	0.271	0.934	0.314	0.273	0.933
knn	embed_lgbm	lgbm	0.351	0.697	0.274	0.274	0.934	0.319	0.274	0.933
knn	wrap	none	0.343	0.694	0.259	0.256	0.933	0.294	0.259	0.932
sgd	wrap	none	0.285	0.667	0.200	0.173	0.928	0.263	0.200	0.925
sgd	none	none	0.230	0.562	0.184	0.161	0.922	0.245	0.184	0.921
sgd	embed_linear	linear	0.227	0.550	0.182	0.155	0.922	0.246	0.182	0.921

sgd	pred	none	0.226	0.554	0.161	0.126	0.922	0.232	0.161	0.920
lr	pred	none	0.217	0.545	0.119	0.069	0.927	0.307	0.119	0.913
lr	embed_ linear	linear	0.217	0.547	0.118	0.068	0.927	0.302	0.118	0.913
lr	none	none	0.217	0.547	0.118	0.068	0.927	0.302	0.118	0.913
lr	assoc	none	0.217	0.547	0.118	0.068	0.927	0.302	0.118	0.913
lr	wrap	none	0.217	0.553	0.118	0.067	0.928	0.284	0.118	0.913
lr	embed_ lgbm	lgbm	0.216	0.547	0.117	0.069	0.927	0.340	0.117	0.913
sgd	assoc	none	0.203	0.553	0.169	0.145	0.919	0.240	0.169	0.918
dummy	assoc	none	0.184	0.500	0.091	0.028	0.918	0.184	0.091	0.909
dummy	embed_ lgbm	lgbm	0.184	0.500	0.091	0.028	0.918	0.184	0.091	0.909
dummy	pred	none	0.184	0.500	0.091	0.028	0.918	0.184	0.091	0.909

dummy	none	none	0.184	0.500	0.091	0.028	0.918	0.184	0.091	0.909
dummy	embed_ linear	linear	0.184	0.500	0.091	0.028	0.918	0.184	0.091	0.909
dummy	wrap	none	0.184	0.500	0.091	0.028	0.918	0.184	0.091	0.909
gandalf	embed_ lgbm	lgbm	0.174	0.611	0.121	0.071	0.917	0.215	0.121	0.914
sgd	embed_ lgbm	lgbm	0.174	0.560	0.128	0.113	0.916	0.160	0.128	0.915
gandalf	none	none	0.109	0.564	0.097	0.039	0.896	0.189	0.097	0.910
gandalf	assoc	none	0.091	0.527	0.097	0.030	0.908	0.078	0.097	0.910
gandalf	embed_ linear	linear	0.077	0.578	0.092	0.036	0.905	0.164	0.092	0.910
gandalf	wrap	none	0.063	0.551	0.084	0.015	0.911	0.067	0.084	0.909
gandalf	pred	none	0.055	0.582	0.095	0.027	0.911	0.145	0.095	0.910

Table: 2.2.3. 5-fold performance on holdout set (PHASE 2)										
Model	**Selection**	**Embed Selector**	**ACC**	**AUROC**	**Bal-ACC**	**F1**	**NPV**	**PPV**	**Sens**	**Spec**
lgbm	none	none	0.957	0.997	0.956	0.956	0.996	0.964	0.956	0.996
sgd	none	none	0.954	0.999	0.954	0.954	0.995	0.960	0.954	0.995
sgd	assoc	none	0.954	0.999	0.954	0.954	0.995	0.960	0.954	0.995
lr	none	none	0.951	0.999	0.951	0.950	0.995	0.958	0.951	0.995
sgd	pred	none	0.951	0.997	0.950	0.950	0.995	0.960	0.950	0.995
lgbm	embed_linear	linear	0.951	0.998	0.951	0.951	0.995	0.957	0.951	0.995
lr	embed_linear	linear	0.948	0.999	0.948	0.947	0.995	0.955	0.948	0.995
lr	assoc	none	0.945	0.999	0.945	0.944	0.995	0.953	0.945	0.995
lr	pred	none	0.945	0.998	0.944	0.944	0.995	0.953	0.944	0.995
lgbm	assoc	none	0.945	0.997	0.945	0.945	0.995	0.951	0.945	0.995
sgd	embed_linear	linear	0.945	0.997	0.945	0.944	0.995	0.952	0.945	0.995

lgbm	pred	none	0.939	0.997	0.938	0.938	0.994	0.949	0.938	0.994
knn	pred	none	0.902	0.990	0.901	0.900	0.991	0.929	0.901	0.990
rf	embed_linear	linear	0.890	0.995	0.890	0.887	0.989	0.906	0.890	0.989
knn	none	none	0.890	0.987	0.888	0.884	0.989	0.921	0.888	0.989
knn	assoc	none	0.890	0.987	0.888	0.884	0.989	0.921	0.888	0.989
knn	embed_linear	linear	0.890	0.987	0.888	0.884	0.989	0.921	0.888	0.989
rf	none	none	0.854	0.973	0.853	0.848	0.986	0.875	0.853	0.985
rf	assoc	none	0.851	0.989	0.850	0.840	0.986	0.882	0.850	0.985
rf	pred	none	0.842	0.984	0.841	0.833	0.985	0.870	0.841	0.984
dummy	none	none	0.101	0.520	0.101	0.097	0.910	0.103	0.101	0.910
dummy	embed_linear	linear	0.091	0.521	0.092	0.089	0.909	0.090	0.092	0.909
dummy	pred	none	0.091	0.500	0.091	0.015	0.909	0.091	0.091	0.909
dummy	assoc	none	0.091	0.500	0.091	0.015	0.909	0.091	0.091	0.909

3. References

1. Xiao, Haixia. 2021. Weather phenomenon database (WEAPD), Version 1.0. Harvard Dataverse, V1. doi:10.7910/DVN/M8JQCR. URL-[https://doi.org/10.7910/DVN/M8JQCR]

2.StFX Executables. 2024. *DF-Analyze: AutoML and Data Analysis Framework*. Version 1.0.0. Retrieved from https://github.com/stfxecutables/df-analyze

3. Hussain, Moayadeldin, Levman, Jacob. 2024. *X-vision-helper: Adjusting Deep Learners for Image Classification Problems with a Single Command*. Version 1.0.0. URL-https://github.com/moayadeldin/X-vision-helper

4. He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *ArXiv*. https://arxiv.org/abs/1512.03385

5. Xiao, Haixia, Zhang, Feng, Shen, Zhongping, Wu, Kun, and Zhang, Jinglin. 2021. *Classification of Weather Phenomenons from Images by Using Deep Convolutional Neural Network.* Earth and Space Science. Published 07 April 2021. doi:10.1029/2020EA001604.URL-[https://doi.org/10.1029/2020EA001604]

——-- END OF THE REPORT —---