SCC451 Coursework Report

Nhan Nguyen
School of Computing and Communications
SCC451 Machine Learning

I. Task 1: Basel Climate Dataset

The Basel Climate Dataset (ClimateDataBasel.csv) is a subset of publicly available weather data from Basel, Switzerland, obtained by Meteoblue[1]. It contains 1,763 records with 18 features collected during the summer and winter seasons from 2010 to 2019.

As the dataset lacks column headers, assigning appropriate names to each feature is necessary to ensure readability, consistency, and efficient data handling during analysis. The complete list of column names and their meanings is provided in Table VI .

A. Preprocessing

Preprocessing is an important step ensure our data is high quality and reliable for analysis. Our process for this step is as follow.

1) Data cleaning: handling missing data, identifying and treating outliers, resolving data inconsistencies and removing duplicates.
2) Feature scaling.
3) Feature selection / extraction.

a) *Data Cleaning:*

First, import data file as DataFrame using pandas package. We named it basel_df.

Handling missing data By using isna() function of Pandas. We can see that the dataset has no missing value. Code for missing value detection is in Fig. 11 (Appendix A).

Feature Scaling Because the dataset includes features in different measurement unit (e.g., temperature in °C, humidity in %, pressure in hPa, precipitation in mm, and wind in km/h), directly compare raw values would bias distance-based clustering algorithm. Therefore, feature scaling is applied to bring all variables to comparable baseline and ensure they contribute equally in learning process.

There are three methods considered, including Z-score standardization, Min-max Normalization and Robust Scaling. Table I shows the benefits and tradeoffs of these methods. Among these, Robust Scaling is the most suitable because of its robustness with skewed distribution and outliers (see code implemented in Fig. 15). All the rest methods are severely impacted by outliers and skewness because they depend on the mean (Z-score standardization) or the min and max of data (Min-max Normalization) [1]. Robust Scaling is a method that use median and interquartile range (IQR) for standardization. For each feature x,

$$x' = \frac{x - \text{median}(x)}{\text{IQR}(x)} \tag{1}$$

where

$$\text{IQR}(x) = Q_3(x) - Q_1(x) \tag{2}$$

The median is less sentitive to outliers than the mean and the IQR measure the spread of the central 50% of data, ignoring extreme values.

TABLE I
Feature Scaling Methods

Method	Advantage	Disadvantages
Z-score Standardization	Centres data to mean 0 and variance 1 Works well for approximately normal distributions. Common choice for distance-based algorithm (e.g., K-Means)	Sensitive to outliers and non-Gaussian data. Not ideal for skewed or heavy-tailed features.
Min-max Normalization	Rescales data to a bounded range [0, 1] Preserves relative relationship among values Useful for distance-based algorithms (e.g., k-Nearest Neighbors)	Extreme sensitive to outliers
Robust Scaling	Robust to outliers and skewed distributions. Maintains feature comparability across mixed-scale data. Suitable for heavy-tailed variables.	not optimal for symmetric, normal distribtion. computationally expensive.

Identifying and treating outliers / noise Outliers are data points that deviate significantly from the overall pattern of the dataset. They can take unusually high or low values compared to the majority of observations, potentially indicating measurement errors, or rare but valid phenomena [2]. In the following step, we will detect potential anomalies across all features and decide on appropriate treatments, such as removal, transformation, or imputation, depending on their impact on the dataset. According to [3], there are four primary categories of outlier dectection model: (1) Neigborhood-based model, (2) Subspace-based model, (3) Ensemble-based model and (4) Mixed-typed model. Here we analyze four methods corresponding to four categories, i.e. Local Outlier Factor (LOF) (Neigborhood-based), Subspace

[1]https://www.meteoblue.com/en/weather/week/galgate_united-kingdom_2648924

Outlier Degree (SOD) (Subspace-based), High Contrast Subspace (HiCF) (Ensemble-based) and Link-Based Outlier and Anomaly Detection (LOADED) (Mix-type). Our dataset has only 1750 rows so it is reasonable to choose the LOF method for detecting anomalies for its efficiency with small dataset (Table II). The LOF algorithm [4] is a neigborhood-based detection method which measure how the local density of a data point differs from that of its neighbors. The main idea is to calculate a score which tell us how much sparser a data point compared to its neigbors. A point in a dense region has a local density similar to its neighbors whereas in a sparse region (outlier), it has a local density lower than its neighbors.

$$\text{LOF Score} \approx \frac{\text{Average Density of Neighbors}}{\text{Density of a Point}} \tag{3}$$

After detecting anomalies with LOF algorithm, a total of 0.74% of samples are marked as outliers. Instead of remove them, which may represent actual rare extreme events, we cap them under its 1st and 99th percentile which is called Winsorization technique. It will limit the effect of extreme values while preserving consistency of the distribution.

TABLE II
ANOMALY DETECTION METHODS

Method / Technique	**Advantages (Strengths)**	**Disadvantages (Limitations)**
Local Outlier Factor (LOF) — Neighborhood-based	Measures how isolated a point is relative to its neighborhood density. Handles variable-density datasets effectively. Can reveal subtle, locally defined anomalies. Work well with small dataset.	Time complexity $O(n^2)$. Not well-suited for large-scale datasets. Requires tuning of neighborhood size and data scaling.
Subspace Outlier Degree (SOD) — Subspace-based	Detects anomalies confined to certain feature subspaces. Effective for correlated, high-dimensional attributes. Useful when global distance metrics are unreliable.	Needs selection of suitable subspaces and thresholds. Computationally demanding for large datasets. May underperform if irrelevant dimensions are included.
High Contrast Subspace (HiCF) — Ensemble-based	Identifies outliers based on contrasting behavior across subspaces. Robust for complex, high-dimensional data. Gains stability through ensemble integration.	Computationally expensive due to repeated subspace analysis. Requires well-defined contrast criteria. Unsuitable for real-time or streaming applications.
Link-Based Outlier and Anomaly Detection (LOADED) — Mix-type / Hybrid	Achieves high detection accuracy and strong true-positive rates. Supports both categorical and numerical attributes. Combines diverse detection mechanisms adaptively.	Complex model structure with several parameters to tune. Computational cost not fully reported but expected to be high. Interpretation can be difficult due to hybrid design.

Feature selection and extraction Feature selection and extraction are two important steps in data preprocessing. The goal of these two is to identify and keep only valuable variables that contribute most to algorithm. By applying appropriate methods, we can build a more robust, simpler and less time-consuming as well as outstanding performance model. Feature selection focuses on selecting most relevant features that necessarily involve in predictive model while drop the rest features. Feature extraction means transforming original features that in high-dimensional into new features with lower-dimensional representation without losing any informative value.

In this project, we use Pearson's product-moment correlation coefficient (PMCC) [5] for finding the correlation between features. The Pearson correlation method is the most commonly used in practice [6]. It measures how strongly two variables change together and in what direction. Given two variables X and Y,

$$r = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} \tag{4}$$

where $\text{cov}(X, Y)$ is the covariance between X and Y and σ_X, σ_Y are the standard deviations of X and Y, respectively. The value of r is in range $[-1, 1]$. The higher in value of r the tighter in relationship between X and Y. As shown in Fig. 1, temperature features (i.e. temp_min_c, temp_mean_c, temp_max_c) are highly correlated with each other. The pressure features (i.e. slp_min_hpa, slp_max_hpa, slp_mean_hpa) are also correlated. Similarly, wind and gust features are correlated. The rest features are weakly correlated. From this heatmap, we can easily identify which features providing similar information and which are unique. It is useful for feature selection and avoiding redundancy. From this information, we can easily remove all high correlated columns (see code implemented and columns removed in Fig. 16).

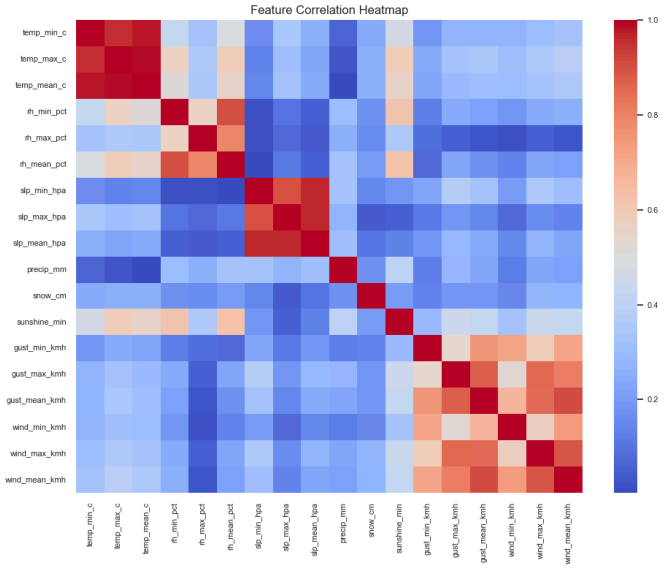

Fig. 1. Correlation between features

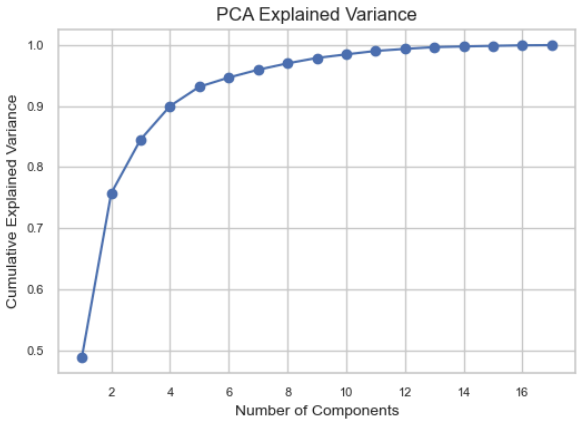

Fig. 2. PCA Explained Variance

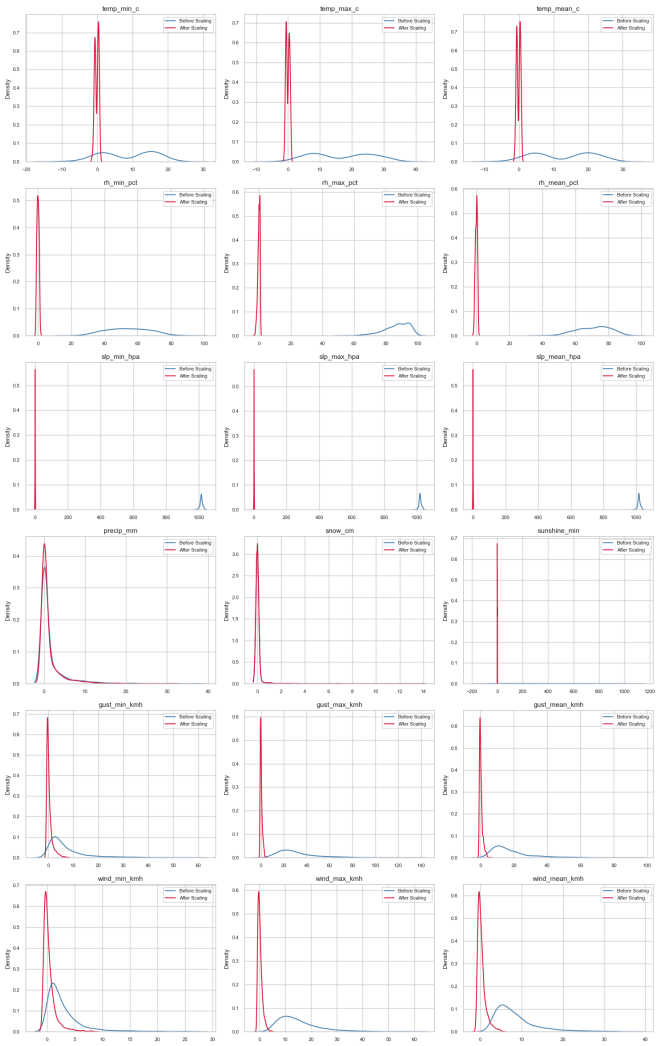

Fig. 3. the differences between before and after scaling. Before scaling, the data distributions are varied among features but after applying Robust Scaling method, we can see mostly distributions now look similar.

We use Principal Component Analysis (PCA) for dimensionality reduction. With n_components = 4, we capture approximately 90% information (Fig. 2). It helps reduce a cost of computation of the model.

B. *Clustering*

According to [7], [8], clustering algorithms can be categorized into seven buckets: (1) Partition-based clustering, (2) Hierarchical clustering, (3) Density-based clustering, (4) Grid-based clustering, (5) Model-based clustering, (6) Graph-based clustering and (7) Deep Learning-based clustering. These models serve in different situations and business based on their characteristics.

To choose suitable algorithms, we look at the structure of the dataset. As shown in Fig. 18, the dataset contains bimodal variables (temperature), heavily skewed variables (precipitation, snow, wind, gust). We can conclude that the climate data is not spherical, not Gaussian and does not have uniform cluster density.

Here we choose three algorithms for clustering task: K-Means from partition-based approach, HDBSCAN from density-based clustering category and GMM from model-based approach.

K-Means K-Means [9], [10] is an unsupervised, partition-based clustering algorithm that divides data into K groups based on similarity. It is one of the simplest, fastest, and most widely used in real world. The main idea of K-Means is to find K cluster centers (centroid) such that each data point belongs to the cluster with closest centroid by minizing the Within-Cluster Sum of Squares (WCSS) which is calculated by summing the squared Euclidean distance between each data point (x_i) and the centroid (μ_i) of the assign cluster (C_k).

$$\text{WCSS} = \sum_{i}^{n} \min_{\mu_j \in C_k} \left\| x_i - \mu_j \right\|_2 \tag{5}$$

In this task, we run K-Means clustering with multiple options of k ranging from 2 to 10 and use Silhouette Score to select the optimal number of clusters. As shown in Fig. 4, the analysis reveals that $k = 4$ provides the best balance between cluster cohesion and separation, achieving a Silhouette Score of 0.4890. This suggests that the climate data naturally divides into four seasonal patterns, potentially representing spring, summer, autumn, and winter weather conditions.

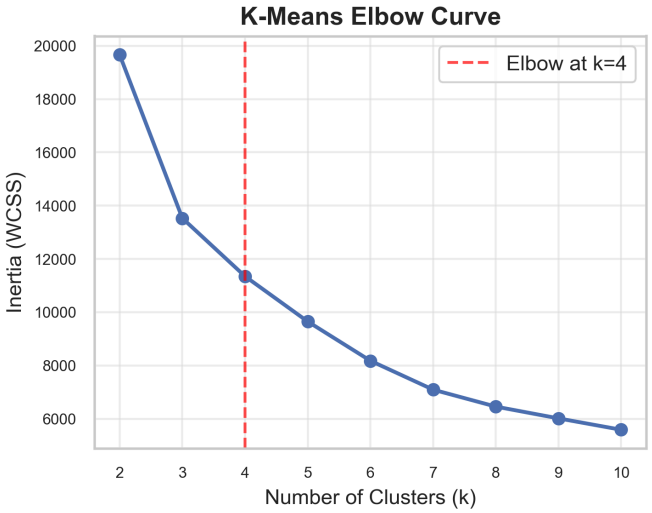

Fig. 4. K-Means elbow curve

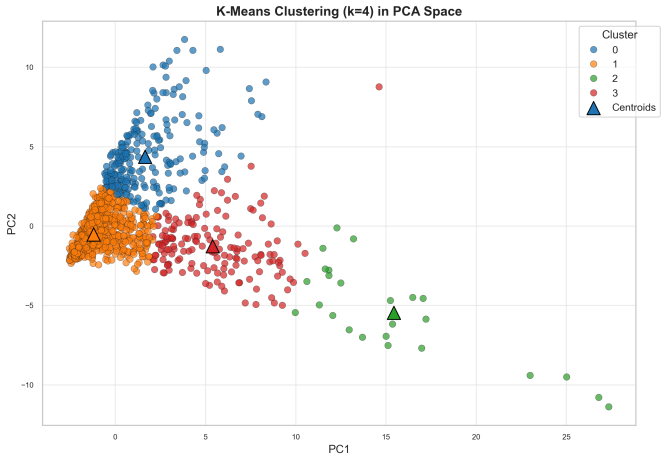

Fig. 5. K-Means Clustering Visualization

HDBSCAN HDBSCAN [11] is an algorithm that groups points based on the density of the surrounding region. Extending from DBSCAN, it builds a hierarchy of density-based clusters and selects the most stable clusters from them. Unlike K-Means, HDBSCAN does not require specifying the number of clusters in advance and can automatically identify outliers as noise points. The algorithm works by computing a minimum spanning tree of the mutual reachability distance, then extracting a hierarchy of clusters based on density thresholds, and finally selecting the most persistent clusters using a stability measure. In this task, we test different min_cluster_size parameters (10, 15, 20, 30, 50) and select the configuration with the highest Silhouette Score. The best configuration uses min_cluster_size=10, identifying 2 main clusters and 275 noise points (15.60% of the data).

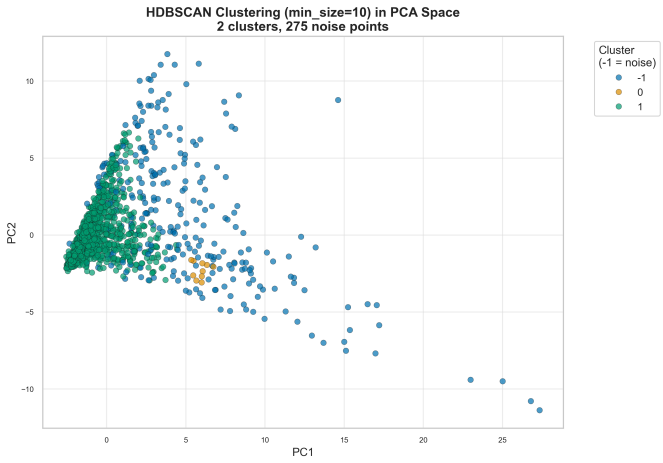

Fig. 6. HDBSCAN Clustering Visualization

GMM Gaussian Mixture Model [12] is a probabilistic, model-based clustering algorithm that assumes data is generated from a mixture of multiple Gaussian distributions. Unlike K-Means which assigns each point to exactly one cluster (hard assignment), GMM provides soft assignments by computing the probability that each point belongs to each cluster. This is particularly useful when cluster boundaries

are uncertain or when points may belong to multiple clusters. The algorithm uses the Expectation-Maximization (EM) algorithm to iteratively estimate the parameters (means, covariances, and mixture weights) of the Gaussian components. For cluster k, the probability that a data point $\boldsymbol{x}$ belongs to that cluster is:

$$P(k \mid \boldsymbol{x}) = \frac{\pi_k \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \tag{6}$$

where π_k is the mixing coefficient, $\boldsymbol{\mu}_k$ is the mean vector, and $\boldsymbol{\Sigma}_k$ is the covariance matrix. We use covariance_type='full' to allow elliptical cluster shapes, which is more appropriate for climate data than spherical clusters. Model selection is performed using Bayesian Information Criterion (BIC) and Akaike Information Criterion (AIC), with lower values indicating better models. As shown in Fig. 7, the best configuration uses 2 components.

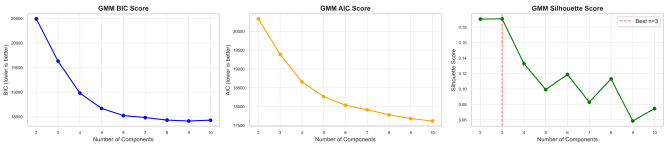

Fig. 7. GMM model selection using BIC, AIC, and Silhouette Score

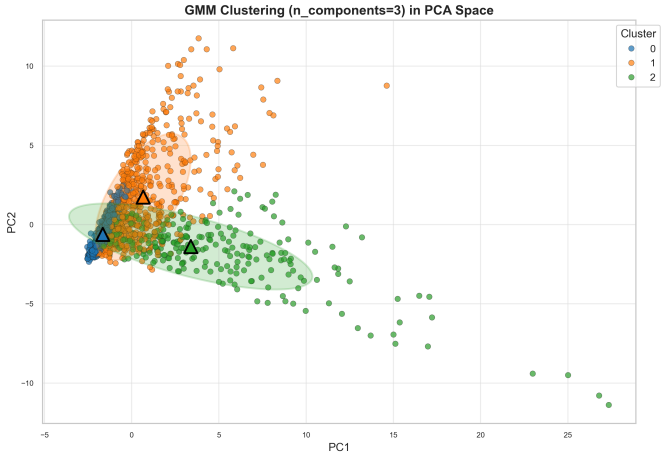

Fig. 8. GMM Clustering Visualization with Gaussian ellipses showing the probabilistic cluster boundaries

a) *Evaluation and Comparison:*

To evaluate the performance of the three clustering algorithms, we use three widely-adopted metrics: Silhouette Score, Davies-Bouldin Index, and Calinski-Harabasz Index. The Silhouette Score measures how similar a point is to its own cluster compared to other clusters, with values ranging from −1 to 1 (higher is better). The Davies-Bouldin Index evaluates the average similarity between each cluster and its most similar cluster, where lower values indicate better separation. The Calinski-Harabasz Index (also known as Variance Ratio Criterion) measures the ratio of between-cluster dispersion to within-cluster dispersion, with higher values indicating better-defined clusters.

TABLE III
COMPARISON OF CLUSTERING ALGORITHMS PERFORMANCE ON BASEL CLIMATE DATASET

Metric	K-Means	HDBSCAN	GMM
Number of Clusters	4	2	3
Silhouette Score	0.4890	0.5704	0.1909
Davies-Bouldin Index	0.8551	0.4255	1.6062
Calinski-Harabasz Score	982.21	85.45	490.63

As shown in Table III, HDBSCAN achieved the best overall performance with the highest Silhouette Score (0.5704) and the lowest Davies-Bouldin Index (0.4255), indicating superior cluster quality and separation. HDBSCAN identified 2 main clusters with 275 noise points (15.60% of the data), suggesting the presence of two dominant climate patterns (summer and winter) along with transitional or extreme weather events . The noise points represent anomalous weather conditions that HDBSCAN appropriately separates rather than forcing into existing clusters, demonstrating the algorithm's robustness to outliers. K-Means, configured with 4 clusters based on silhouette analysis, achieved a moderate Silhouette Score (0.4890) and the highest Calinski-Harabasz Score (982.21), indicating well-defined cluster centers. However, its Davies-Bouldin score (0.8551) was higher than HDBSCAN, suggesting more overlap between clusters. This is expected given K-Means' assumption of spherical clusters and its sensitivity to the bimodal and skewed distributions present in climate data. GMM, with 3 components, performed poorly compared to the other algorithms, achieving the lowest Silhouette Score (0.1909) and the highest Davies-Bouldin Index (1.6062), indicating poorly separated clusters with significant overlap.

The choice of algorithm depends on the specific analytical goals. HDBSCAN is the most suitable for this climate dataset, effectively identifying the two main seasonal patterns while robustly handling outliers and extreme weather events. The results suggest that the Basel climate data exhibits two primary seasonal clusters with a substantial number of transitional or anomalous days, validating the use of density-based clustering approaches for meteorological data analysis.

II. Task 2: Image Processing with Deep Neural Networks (DNNs)

The purpose of this task is to effectively, thoroughly apply pre-trained DNNs for image clustering and classification tasks. In this task, we choose two datasets, i.e. Kaggle: Cats vs Dogs Dataset [13] and Food101 [14]. Kaggle: Cats vs Dogs Dataset is a dataset providing a collection of photos about cats and dogs. Statistically, there are 12491 pictures of cat and 12470 images of dog. Food101 includes images of food. With over 100000 files, it is a good set of data for image clustering and classification training. We also choose DINOV2 [15] (dinov2_vits14 model via Pytorch Hub) and DenseNet [16] (DenseNet-121 model pre-trained on ImageNet) as our DNN models. DINOV2, developed by Meta AI Research (or FAIR), is a self-supervised vision model trained using self-distilla-

tion and self-supervised learning which means it can learn from unlabeled images. DenseNet is a CNN architecture, well-known for its densen connectivity pattern, where each layers receives the feature maps of all previous layers as input.

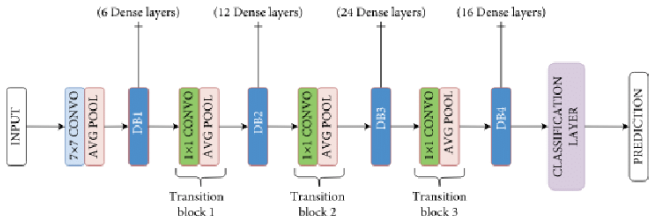

Fig. 9. Illutration of DenseNet architecture. Image taken from [17].

A. Data Preprocessing

In this project, we resizes images to 256×256 using interpolation mode Bilinear, followed by a center crop of 224. After that, they will be normalized using mean $= [0.485, 0.456, 0.406]$ and std $= [0.229, 0.224, 0.225]$. These values are industry-standard and implemented in torchvision's preset for classification.[2]

B. Feature Extraction

Image feature extraction is a process of converting an image into a set of meaningful numerical features that capture important visual information such as shapes, textures, colors. In this task, we use deep learning-based model for extracting features. Deep learning-based feature extraction. Deep neural networks (i.e. CNN and Transformers) is suitable for this step because of their ability to learn hierarchical, sementically rich representations directly from pixels. A DNN informative outputs are essential and useful for downstream processes like clustering or classification.

As you can see in Fig. 9, the last layer of DenseNet-121 is the classification layer. Thus, to use the model for feature extraction, we need to drop the last layer. With DINOV2, we can use it as normal. After the process of extracting feature is done, we save all results into .npy files for later use.

The same as Section I, we use PCA technique for dimensionality reduction. We retain 95% of the variance, reducing the Food101 feature vectors from 1024 to 14 dimensions with DenseNet-121 and from 384 to 202 dimensions with DINOv2. Regarding Cats vs Dogs dataset, we reduce feature vectors from 1024 to 14 dimensions with DenseNet-121 and from 384 to 197 dimensions with DINOV2 (shown in Table IV).

TABLE IV
BEFORE AND AFTER REDUCTION DIMENSIONS OF TWO DATASETS WITH TWO MODELS

Dataset - Model	Original Dimensions	Reduced Dimensions
Food101 - DenseNet-121	1024	14
Food101 - DINOV2	384	202
Cats vs Dogs - DenseNet-121	1024	14
Cats vs Dogs - DINOV2	384	197

C. Clustering

In clustering task, we use K-Means clustering method. $k = 101$ and $k = 2$ are used for Food101 dataset and Cats vs. Dgos dataset, respectively.

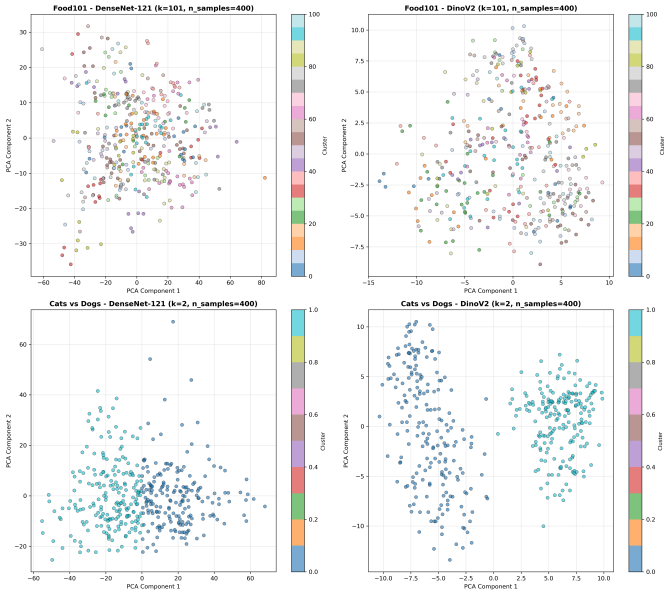

Fig. 10. K-Means Cluster of all combinations of datasets and models. We use 400 samples for visualization.

Dataset - Model	k	Silhouette score	Davies-Bouldin Index	Calinski-Harabasz Score
Food101 - DenseNet-121	101	0.0986	1.6199	5728.69
Food101 - DINOV2	101	0.0904	2.6753	888.37
Cats vs Dogs - DenseNet-121	2	0.3468	1.0796	18609.98
Cats vs Dogs - DINOV2	2	0.0973	3.0443	2671.85

[2]https://github.com/pytorch/vision/blob/main/references/classification/presets.py#L47-L69

References

[1] L. B. de Amorim, G. D. Cavalcanti, and R. M. Cruz, "The choice of scaling technique matters for classification performance," *Appl. Soft Comput.*, vol. 133, no. C, Jan. 2023, doi: 10.1016/j.asoc.2022.109924.

[2] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

[3] C. Das, A. Dubey, and A. Rasool, "Outlier Detection Techniques: A Comparative Study," *Edge Analytics*, vol. 869. Springer Singapore, Singapore, pp. 551–566, 2022. doi: 10.1007/978-981-19-0019-8_42.

[4] D. S. Upadhyaya and K. Singh, "Nearest Neighbour Based Outlier Detection Techniques," 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:212537073

[5] M.-T. Puth, M. Neuhäuser, and G. D. Ruxton, "Effective Use of Pearson's Product–Moment Correlation Coefficient," *Animal Behaviour*, vol. 93, pp. 183–189, July 2014, doi: 10.1016/j.anbehav.2014.05.003.

[6] D. Nettleton, "Chapter 6. Selection of Variables and Factor Derivation," 2014, p. . doi: 10.1016/B978-0-12-416602-8.00006-6.

[7] H. Yin, A. Aryani, S. Petrie, A. Nambissan, A. Astudillo, and S. Cao, "A Rapid Review of Clustering Algorithms," no. arXiv:2401.07389. arXiv, Jan. 2024. doi: 10.48550/arXiv.2401.07389.

[8] A. A. Wani, "Comprehensive Analysis of Clustering Algorithms: Exploring Limitations and Innovative Solutions," *PeerJ Computer Science*, vol. 10, p. e2286, Aug. 2024, doi: 10.7717/peerj-cs.2286.

[9] S. Lloyd, "Least squares quantization in PCM," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[10] J. MacQueen, "Multivariate observations," in *Proceedings ofthe 5th Berkeley Symposium on Mathematical Statisticsand Probability*, 1967, pp. 281–297.

[11] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-Based Clustering Based on Hierarchical Density Estimates," *Advances in Knowledge Discovery and Data Mining*, vol. 7819. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 160–172, 2013. doi: 10.1007/978-3-642-37456-2_14.

[12] D. Reynolds, "Gaussian Mixture Models," in *Encyclopedia of Biometrics*, S. Z. Li and A. Jain, Eds., Boston, MA: Springer US, 2009, pp. 659–663. doi: 10.1007/978-0-387-73003-5_196.

[13] "Cats vs Dogs." [Online]. Available: https://www.kaggle.com/datasets/karakaggle/kaggle-cat-vs-dog-dataset

[14] "Food101." [Online]. Available: https://www.kaggle.com/datasets/dansbecker/food-101

[15] M. Oquab *et al.*, "DINOv2: Learning Robust Visual Features without Supervision." 2023.

[16] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely Connected Convolutional Networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2016, [Online]. Available: https://api.semanticscholar.org/CorpusID:9433631

[17] G. Sujawat and A. Kumar, "A Deep Learning-Based Methodology For Plant Disease Identification And Diagnosis." p. , 2022. doi: 10.13140/RG.2.2.12482.66241.

Appendix A: Code

```
[5]
✓ 0s
# Handling missing data
columns = basel_df.columns

missing_summary = pd.DataFrame({
    "Missing Values": basel_df.isna().sum(),
    "Percentage": basel_df.isna().mean() * 100
}).sort_values(by="Missing Values")

missing_summary
```

	Missing Values	Percentage
temp_min_c	0	0.0
temp_max_c	0	0.0
temp_mean_c	0	0.0
rh_min_pct	0	0.0
rh_max_pct	0	0.0
rh_mean_pct	0	0.0
slp_min_hpa	0	0.0
slp_max_hpa	0	0.0
slp_mean_hpa	0	0.0
precip_mm	0	0.0
snow_cm	0	0.0
sunshine_min	0	0.0
gust_min_kmh	0	0.0
gust_max_kmh	0	0.0
gust_mean_kmh	0	0.0
wind_min_kmh	0	0.0
wind_max_kmh	0	0.0
wind_mean_kmh	0	0.0

Fig. 11. Code and result of detecting missing value.

```
# Modified Z-score outlier detection
def modified_zscore(series):
    median = np.mean(series)
    mad = median_abs_deviation(series, nan_policy='omit')
    if mad == 0:
        return np.zeros_like(series)
    return 0.6745 * (series - median) / mad

mad_scores = pd.DataFrame({
    col: modified_zscore(basel_df[col]) for col in num_cols
})

# flag outliers |M| > 3.5
outlier_flags = (np.abs(mad_scores) > 3.5)

total_outliers = outlier_flags.sum().sum()
rows_with_outliers = (outlier_flags.sum(axis=1) > 0).sum()

print(f"Total flagged values: {total_outliers}")
print(f"Rows containing outliers {rows_with_outliers} ({rows_with_outliers/len(basel_df)*100:.2f}%
✓ 0.0s                                                                    Python
Total flagged values: 477
Rows containing outliers 181 (10.27%)
```

Fig. 12. Modified Z-score code and result

```
from sklearn.neighbors import LocalOutlierFactor

lof = LocalOutlierFactor(n_neighbors=20)
labels = lof.fit_predict(basel_df_scaled)
scores = -lof.negative_outlier_factor_

basel_df['lof_label'] = labels
basel_df['lof_score'] = scores

n_outliers = (labels == -1).sum()
print(f"Detected {n_outliers} outliers ({n_outliers/len(basel_df)*100:.2f}%)")
✓ 0.0s
Detected 13 outliers (0.74%)
```

Fig. 13. Local Outlier Factor (LOF)

```
# cap outliers
def cap_outliers(df, lower=0.01, upper=0.99):
    l, u = df.quantile([lower, upper])
    return np.clip(df, l, u)

outlier_mask = basel_df_scaled['lof_score'] == -1
df_treated = basel_df_scaled.copy()
for col in num_cols:
    df_treated.loc[outlier_mask, col] = cap_outliers(basel_df_scaled[col])[outlier_mask]

df_treated.drop(columns=['lof_label', 'lof_score'])
```

Fig. 14. Capping extreme values

```
scaler = RobustScaler()
basel_df_scaled = pd.DataFrame(
    scaler.fit_transform(basel_df[num_cols]),
    columns=num_cols,
    index=basel_df.index
)
✓ 0.0s
```

Fig. 15. Feature Scaling

```
# keep only upper triangle because correlation matrix is symmetric
# that every pair is shown twice, it might be a problem because it
# can lead to accidentially remove both features.
upper = corr.where(np.triu(np.ones(corr.shape), k=1).astype(bool))
high_corr = [col for col in upper.columns if any(upper[col] >= 0.95)]
print("Highly correlated columns to drop:", high_corr)

df_reduced = basel_df_scaled.drop(columns=high_corr)
✓ 0.0s                                                                    Python
Highly correlated columns to drop: ['temp_max_c', 'temp_mean_c', 'slp_mean_hpa']
```

Fig. 16. Remove high correlated columns

```
# Initialize DenseNet121 model (pretrained on ImageNet)
densenet_model = models.densenet121(pretrained=True)
# Remove the classification layer to get feature vectors
densenet_model = nn.Sequential(*list(densenet_model.children())[:-1])
densenet_model = densenet_model.to(device)
densenet_model.eval()

densenet_transform = transforms.Compose([
    transforms.Resize(256, interpolation=transforms.InterpolationMode.BILINEAR),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
✓ 0.2s                                                                    Python
/Users/nhanncv/miniforge3/envs/scc451/lib/python3.13/site-packages/torchvision/models/_uti
  warnings.warn(
/Users/nhanncv/miniforge3/envs/scc451/lib/python3.13/site-packages/torchvision/models/_uti
  warnings.warn(msg)


dinov2_model = torch.hub.load('facebookresearch/dinov2', 'dinov2_vits14')
dinov2_model = dinov2_model.to(device)
dinov2_model.eval()

dinov2_transform = transforms.Compose([
    transforms.Resize(256, interpolation=transforms.InterpolationMode.BILINEAR),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
✓ 0.9s                                                                    Python
Using cache found in /Users/nhanncv/.cache/torch/hub/facebookresearch_dinov2_main
/Users/nhanncv/.cache/torch/hub/facebookresearch_dinov2_main/dinov2/layers/swiglu_ffn.py:5
  warnings.warn("xFormers is not available (SwiGLU)")
/Users/nhanncv/.cache/torch/hub/facebookresearch_dinov2_main/dinov2/layers/attention.py:33:
  warnings.warn("xFormers is not available (Attention)")
/Users/nhanncv/.cache/torch/hub/facebookresearch_dinov2_main/dinov2/layers/block.py:40: Use
  warnings.warn("xFormers is not available (Block)")
```

Fig. 17. Model import and images preprocessing using model transform.

Appendix B: Tables, Figures and Algorithms

TABLE VI
Table columns name and their meaning

Column Name	Description	Unit
temp_min_c	Minimum daily temperature	°C
temp_max_c	Maximum daily temperature	°C
temp_mean_c	Mean daily temperature	°C
rh_min_pct	Minimum daily relative humidity	%
rh_max_pct	Maximum daily relative humidity	%
rh_mean_pct	Mean daily relative humidity	%
slp_min_hpa	Minimum daily sea level pressure	hPa
slp_max_hpa	Maximum daily sea level pressure	hPa
slp_mean_hpa	Mean daily sea level pressure	hPa
precip_mm	Total daily precipitation	mm
snow_cm	Total daily snowfall	cm
sunshine_min	Total sunshine duration per day	min
gust_min_kmh	Minimum daily wind gust speed	km/h
gust_max_kmh	Maximum daily wind gust speed	km/h
gust_mean_kmh	Mean daily wind gust speed	km/h
wind_min_kmh	Minimum daily wind speed	km/h
wind_max_kmh	Maximum daily wind speed	km/h
wind_mean_kmh	Mean daily wind speed	km/h

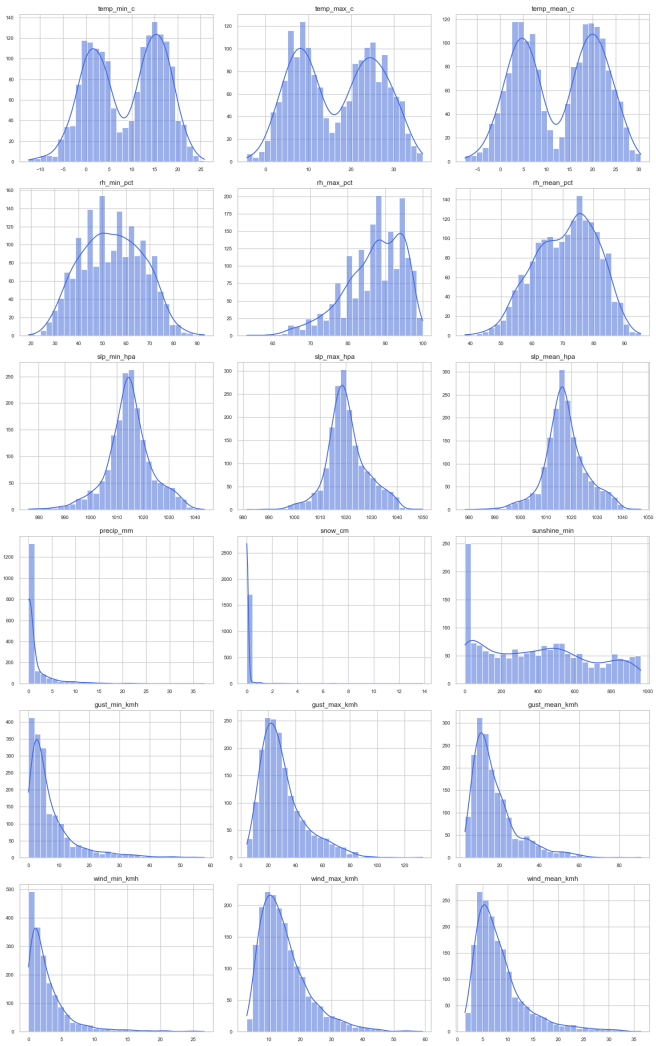

Fig. 18. Distribution of all features