



STAT 576: PHISHING URL PROJECT

CORY SUZUKI, NATE TALAMPAS, RICHARD DIAZDELEON

Intro

- Purpose/Motivation: According to the University of Florida, "Spear phishing is a more personalized, sophisticated, and invasive form of phishing, tailored to steal sensitive information from specific individuals or groups. It's a growing threat—**while spear phishing makes up less than 0.1% of all emails sent, it is responsible for 66% of all data breaches**" (University of Florida, 2024).



Avoid Phishing

Phishing is when an attacker sends someone a fake email hoping to get that person to perform some type of action such as clicking on a malicious link, opening a malicious attachment, or getting the individual to enter sensitive information into a fake website. Phishing emails can lead to loss of research and sensitive data, identity theft, financial damage and more. Phishing emails are very common and it is important for us to know how to protect ourselves against them.

If you suspect that you have received a phishing email, report it to spam@uci.edu




ORDER OF BUSINESS (Aka: The Game Plan!)


- ABOUT THE DATA (Richard)
- DATA EXTRACTION / EDA (Richard)
- FEATURE SELECTION/DIMENSIONALITY REDUCTION (Nate)
- CLUSTERING ALGORITHMS (Cory)
- ANALYSIS RESULTS. FUTURE WORK, & CLOSING REMARKS (Cory & Everyone)



ABOUT THE DATA



[Datasets](#)[Contribute Dataset](#)[About Us](#)



PhiUSIIL Phishing URL (Website)

Donated on 3/3/2024

PhiUSIIL Phishing URL Dataset is a substantial dataset comprising 134,850 legitimate and 100,945 phishing URLs. Most of the URLs we analyzed, while constructing the dataset, are the latest URLs. Features are extracted from the...

Dataset Characteristics

Tabular

Subject Area

Computer Science

Associated Tasks

Classification

Feature Type


Real, Categorical, Integer


Instances

235795


Features


54

 **DOWNLOAD** (14.7 MB)

 **IMPORT IN PYTHON**

CITE

 1 citations

 24982 views

Citations/Acknowledgements

If you use this dataset, please cite:
Prasad, A., & Chandra, S. (2023).

```
{'uci_id': 967, 'name': 'PhiUSIIL Phishing URL (Website)', 'repository_url': 'https://archive.ics.uci.edu/dataset/967/phiusiil+phishing+url+dataset', 'data_url': 'https://archive.ics.uci.edu/static/public/967/data.csv', 'abstract': 'PhiUSIIL Phishing URL Dataset is a substantial dataset comprising 134,850 legitimate and 100,945 phishing URLs. Most of the URLs we analyzed, while constructing the dataset, are the latest URLs. Features are extracted from the source code of the webpage and URL. Features such as CharContinuationRate, URLTitleMatchScore, URLCharProb, and TLDLegitimateProb are derived from existing features.', 'area': 'Computer Science', 'tasks': ['Classification'], 'characteristics': ['Tabular'], 'num_instances': 235795, 'num_features': 54, 'feature_types': ['Real', 'Categorical', 'Integer'], 'demographics': [], 'target_col': ['label'], 'index_col': None, 'has_missing_values': 'no', 'missing_values_symbol': None, 'year_of_dataset_creation': 2024, 'last_updated': 'Sun May 12 2024', 'dataset_doi': 'https://doi.org/10.1016/j.cose.2023.103545', 'creators': ['Arvind Prasad', 'Shalini Chandra'], 'intro_paper': {'ID': 411, 'type': 'NATIVE', 'title': 'PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning', 'authors': 'Arvind Prasad and Shalini Chandra', 'venue': 'Computers & Security', 'year': 2024, 'journal': None, 'DOI': None, 'URL': 'https://doi.org/10.1016/j.cose.2023.103545', 'sha': None, 'corpus': None, 'arxiv': None, 'mag': None, 'acl': None, 'pmid': None, 'pmcid': None}, 'additional_info': {'summary': None, 'purpose': None, 'funded_by': None, 'instances_represent': 'URLs and their corresponding webpages', 'recommended_data_splits': None, 'sensitive_data': None, 'preprocessing_description': None, 'variable_info': 'Column "FILENAME" can be ignored.', 'citation': 'Prasad, A., & Chandra, S. (2023). PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning. Computers & Security, 103545. doi: https://doi.org/10.1016/j.cose.2023.103545'}}
```

	name	role	type	demographic	description \
0	FILENAME	Other	Categorical	None	None
1	URL	Feature	Categorical	None	None
2	URLLength	Feature	Integer	None	None
3	Domain	Feature	Categorical	None	None
4	DomainLength	Feature	Integer	None	None
5	IsDomainIP	Feature	Integer	None	None
6	TLD	Feature	Categorical	None	None
7	URLSimilarityIndex	Feature	Integer	None	None
8	CharContinuationRate	Feature	Integer	None	None
9	TLDLegitimateProb	Feature	Continuous	None	None
10	URLCharProb	Feature	Continuous	None	None
11	TLDLength	Feature	Integer	None	None
12	NoOfSubDomain	Feature	Integer	None	None
..

• Summarization of the Phishing Website URL Dataset:

- **Number of Observations:** 235,795 instances.
- **Number of Features:** 55 features.
- **Feature Description:**
 - * Presence of IP addresses in URLs.
 - * URL length.
 - * Use of "@" symbols.
 - * SSL certificate status.
- **Purpose:** To support the development and evaluation of machine learning models for distinguishing legitimate and phishing websites.
- **Source:** UCI Machine Learning Repository.
- **Year Compiled:** 2015 by Rami Mohammad and Lee McCluskey.

DATA EXTRACTION/ EDA

```
In [9]: import pandas as pd

# Assuming X is your DataFrame
categorical_features = X.select_dtypes(include=['object', 'category']).columns
print(categorical_features)
```

```
Index(['URL', 'Domain', 'TLD', 'Title'], dtype='object')
```

```
In [10]: X_categorical = X.select_dtypes(include=['object', 'category'])
X_categorical
```

Out[10]:		URL	Domain	TLD	Title
0	https://www.southbankmosaics.com	www.southbankmosaics.com	com		à,,áì^à,,²à,,§à,,²à,, à,,áì^à,,²à,,§à,,±à,,¹²à,,¹²à,,µ...
1	https://www.uni-mainz.de	www.uni-mainz.de	de		johannes gutenbergs universitÄt mainz
2	https://www.voicefmradio.co.uk	www.voicefmradio.co.uk	uk		voice fm southampton
3	https://www.sfnmjournal.com	www.sfnmjournal.com	com		home page: seminars in fetal and neonatal medi...
4	https://www.rewildingargentina.org	www.rewildingargentina.org	org		fundaciÃ³n rewilding argentina
...
235790	https://www.skincareliving.com	www.skincareliving.com	com		skincareliving
235791	https://www.winchester.gov.uk	www.winchester.gov.uk	uk		winchestergov
235792	https://www.nononsensedesign.be	www.nononsensedesign.be	be		nononsensedesign
235793	https://patient-cell-40f5.updatedlogmylogin.wo...	patient-cell-40f5.updatedlogmylogin.dev	dev		patient-cell-40f5updatedlogmyloginworkers
235794	https://www.alternativefinland.com	www.alternativefinland.com	com		alternativefinland

235795 rows x 4 columns

```
In [14]: from sklearn.feature_selection import VarianceThreshold

# applying variance threshold
thresholder = VarianceThreshold(threshold=0.1)
X_high_variance = thresholder.fit_transform(X_numeric)

# Convert the numpy array back to a DataFrame
# Retrieve the retained feature indices
features = thresholder.get_support(indices=True)
X_high_variance_df = pd.DataFrame(X_high_variance, columns=[X_numeric.columns[i] for i in features])

print("Shape of data after Variance Threshold:", X_high_variance_df.shape)
X_high_variance_df.head()
```

```
Shape of data after Variance Threshold: (235795, 36)
```

Out[14]:	URLLength	DomainLength	URLSimilarityIndex	TLDLength	NoOfSubDomain	NoOfObfuscatedChar	NoOfLettersInURL	NoOfDe
0	31.0	24.0	100.0	3.0	1.0	0.0	18.0	
1	23.0	16.0	100.0	2.0	1.0	0.0	9.0	
2	29.0	22.0	100.0	2.0	2.0	0.0	15.0	
3	26.0	19.0	100.0	3.0	1.0	0.0	13.0	
4	33.0	26.0	100.0	3.0	1.0	0.0	20.0	

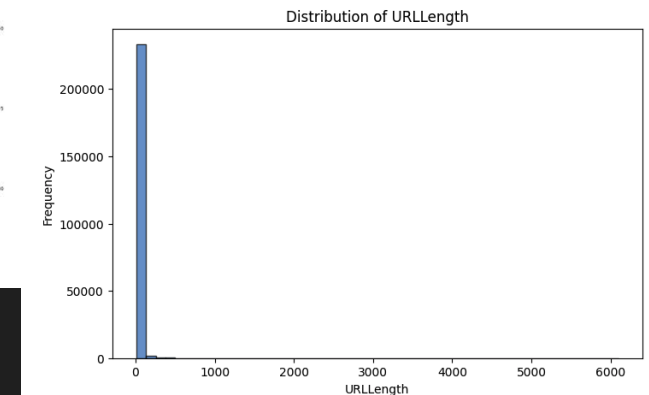
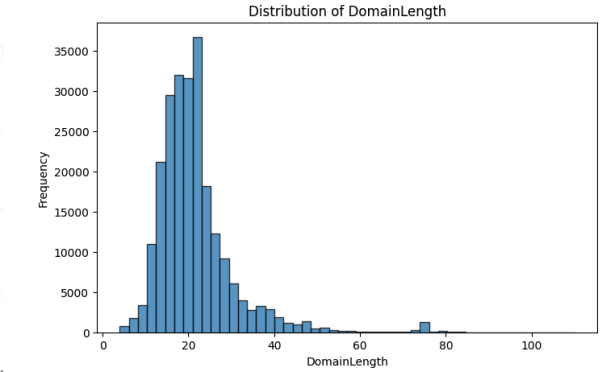
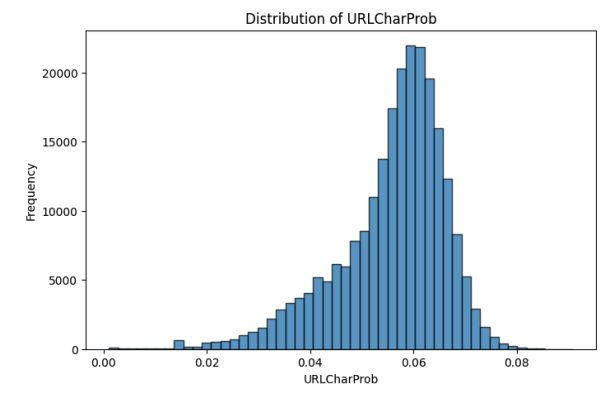
5 rows x 36 columns

```
In [15]: X_high_variance_df.columns
```

```
Out[15]: Index(['URLLength', 'DomainLength', 'URLSimilarityIndex', 'TLDLength',
               'NoOfSubDomain', 'NoOf0bfuscatedChar', 'NoOfLettersInURL',
               'NoOfDigitsInURL', 'NoOfEqualsInURL', 'NoOfAmpersandInURL',
               'NoOf0therSpecialCharsInURL', 'IsHTTPS', 'LineOfCode',
               'LargestLineLength', 'HasTitle', 'DomainTitleMatchScore',
               'URLTitleMatchScore', 'HasFavicon', 'Robots', 'IsResponsive',
               'NoOfURLRedirect', 'HasDescription', 'NoOfPopup', 'NoOfIFrame',
               'HasSocialNet', 'HasSubmitButton', 'HasHiddenFields', 'Bank', 'Pay',
               'HasCopyrightInfo', 'NoOfImage', 'NoOfCSS', 'NoOfJS', 'NoOfSelfRef',
               'NoOfEmptyRef', 'NoOfExternalRef'],
              dtype='object')
```

Data is selected by object and category type. Encountered an error due to the shape of data type since we have a memory capacity only up to 51.7 GiB(run cap). Alternatively, would be to run on the cloud.

- The distribution is unimodal and skewed left, concentrated around 0.06. This suggests that a high probability of specific URL characteristics are legitimate.
- Most domains fall between 15-30 characters
- The majority of URLs have 1 or 2 subdomains



```
Pairs with high correlation (|correlation| > 0.85):
URLLength and NoOfLettersInURL: 0.96
DomainTitleMatchScore and URLTitleMatchScore: 0.96
```

FEATURE SELECTION

- Feature selection by correlation between input features was employed, removing highly correlated features with a correlation coefficient above threshold of 0.85. This step ensured that redundant features with correlations close to 1 were excluded, minimizing multicollinearity and simplifying the feature set for downstream predictive modeling tasks.

```
Pairs with high correlation ( $|correlation| > 0.85$ ):  
URLLength and NoOfLettersInURL: 0.96  
DomainTitleMatchScore and URLTitleMatchScore: 0.96
```

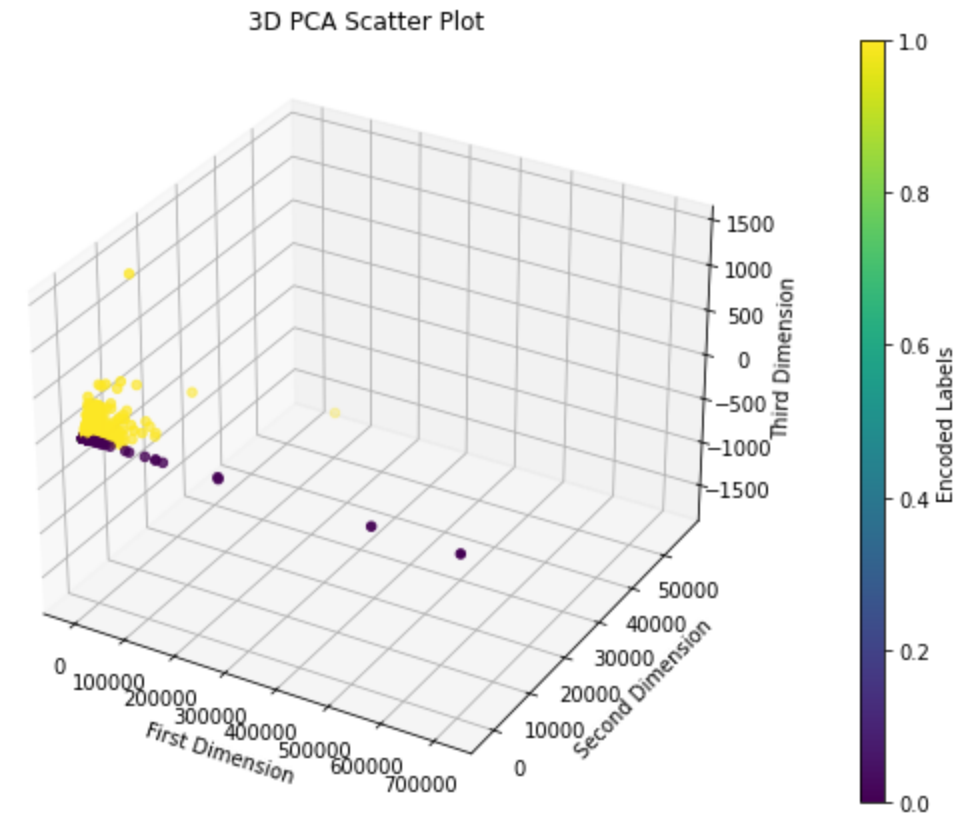
- Feature selection using a variance threshold of 0.1 was also used to remove low variance features that could not distinguish the target variable effectively. Additionally, we removed URL and Domain, as they were deemed unique identifiers and provided little value for generalization
- After feature selection, we were left with 34 feature variables and the target.

DIMENSIONALITY REDUCTION

- The dataset presents a high-dimensional feature space that necessitates dimensionality reduction to simplify analysis, reduce redundancy, and enhance computational efficiency. Both linear and nonlinear methods are employed to preserve essential patterns while projecting the data into a lower-dimensional space.
- Linear methods often struggle to capture complex, nonlinear relationships in high dimensional data, which prompts the need for nonlinear techniques.

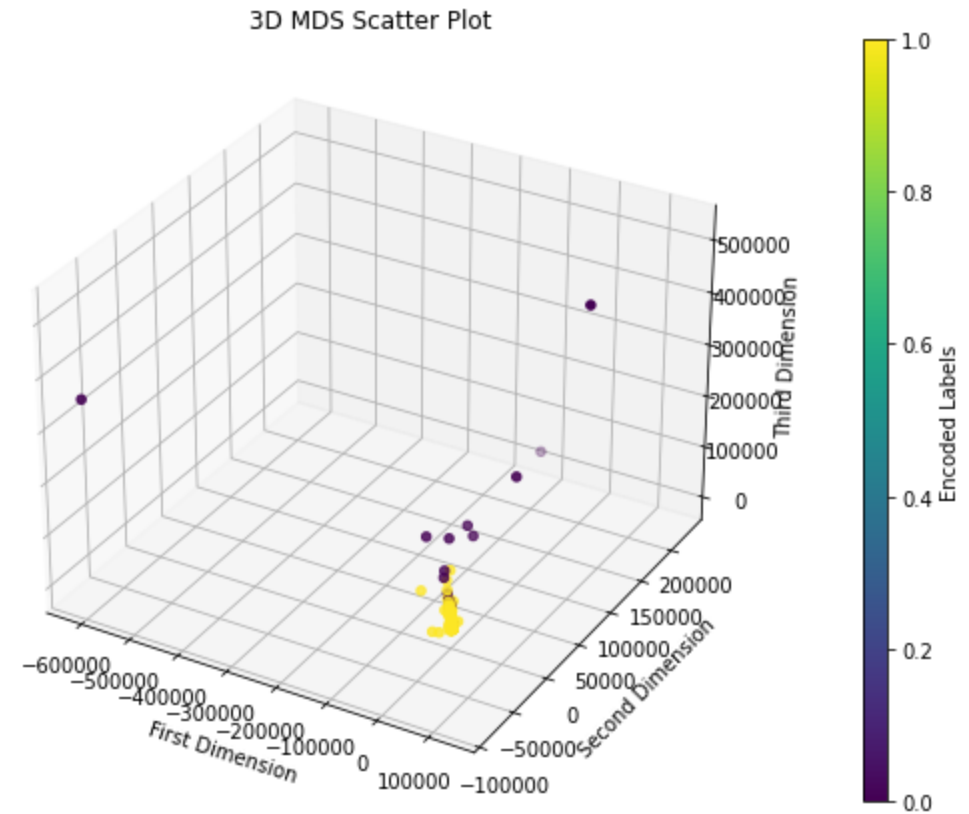
Principal Component Analysis (PCA)

- PCA generates a covariance matrix from the centered data, which emphasizes the covariance between the features. The process involves performing singular value decomposition (SVD) to extract eigenvalues and eigenvectors. The principal components correspond to the eigenvectors associated with the k largest eigenvalues.



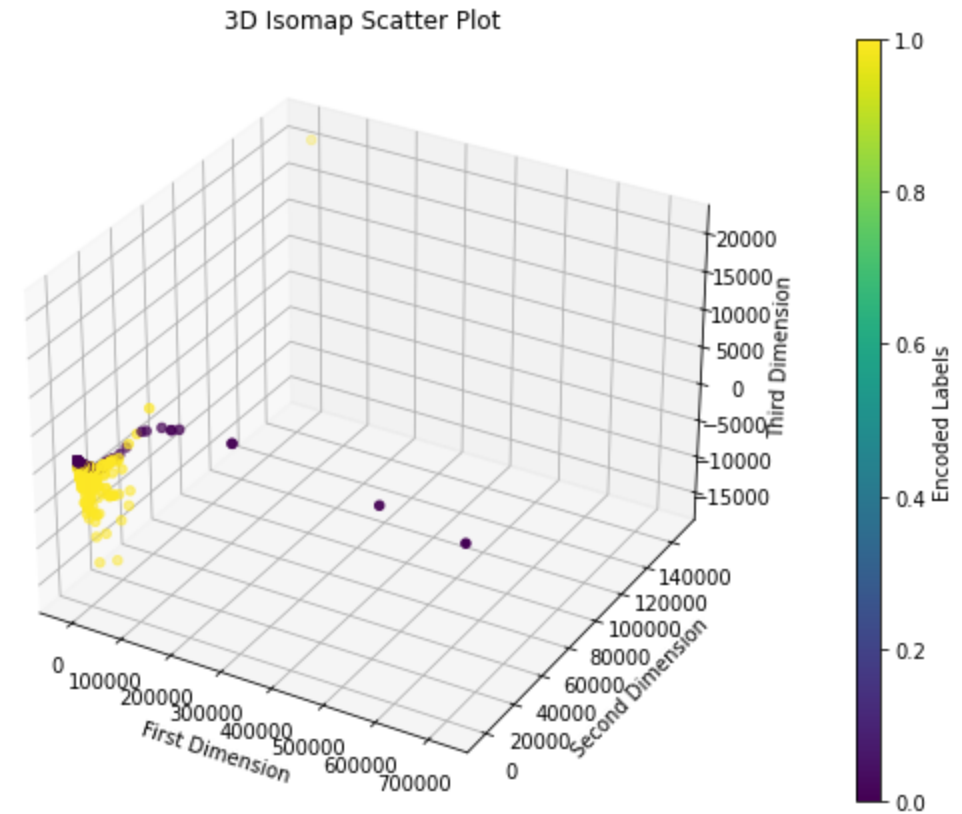
Multidimensional Scaling (MDS)

- MDS aims to find a low-dimensional representation of the data while preserving the pairwise distance between points. The process involves generating a pairwise distance matrix from the data and converting it into a Gram matrix or inner product matrix. By performing SVD on this matrix, we obtain the eigenvalues and eigenvectors and select the k largest eigenvalues to construct the low-dimensional embedding.



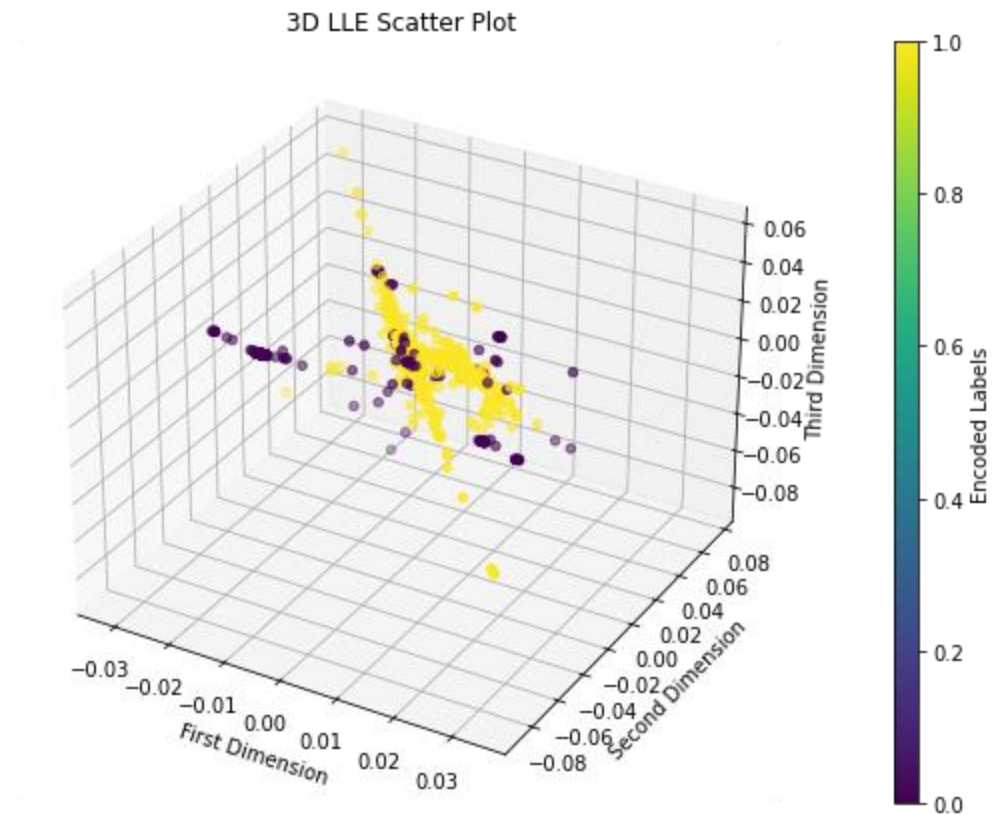
Isomap

- Isomap aims to find the optimal lower-dimensional representations that preserve the global geodesic distances between the data points on a manifold. Isomap relies on geodesic distances, computed using a k-nearest-neighbor graph. This approach holds the assumption that the data is sufficiently large and well-distributed to reveal the underlying manifold structure. This algorithm is effective for manifold-based data but may struggle with outliers or manifolds with high curvature or large holes.



Locally Linear Embedding (LLE)

- LLE assumes that the data points and their neighbors are in a local linear space. This method approximates each data point as a linear combination of its neighbors, using weights that minimize the reconstruction error. The process involves calculating the weight from neighbors. After fixing the weights, the low-dimensional vectors are determined by minimizing its cost function, which can then be utilized to map the embedded coordinates into a lower-dimensional space. This process has limitations, such as the lack of an out-of-sample extension and limited theoretical guidance about the number of intrinsic dimensions.

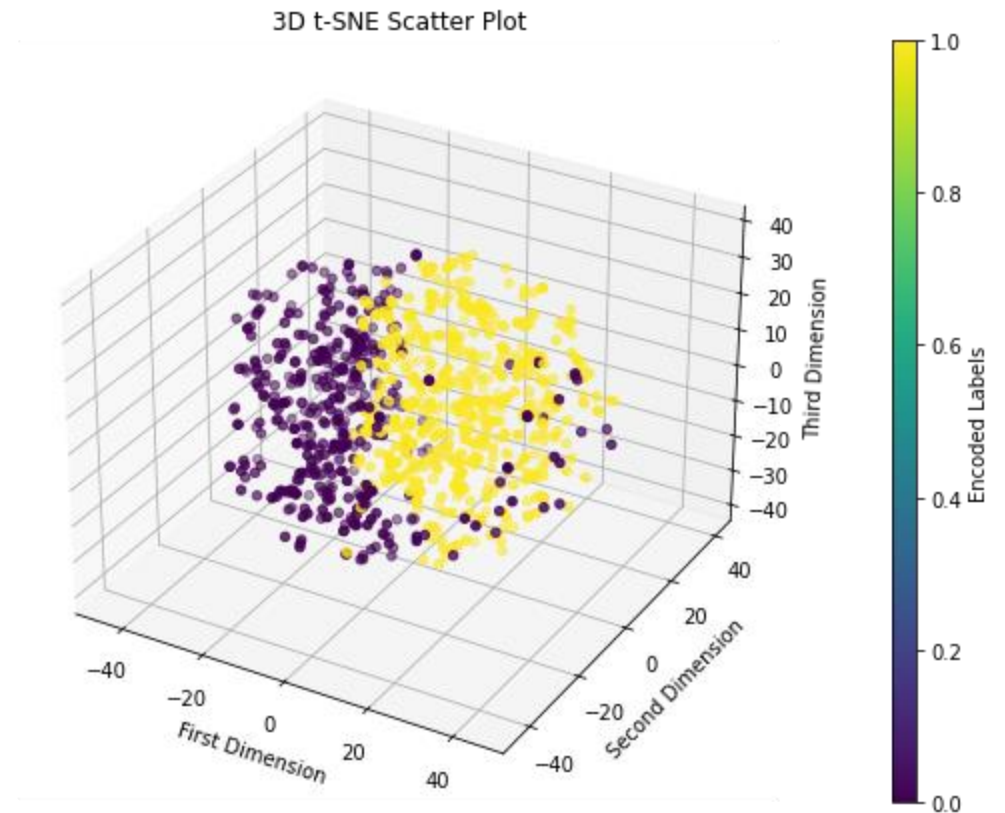


t-distributed Stochastic Neighbor Embedding (t-SNE)

- t-SNE is designed to visualize high-dimensional data by preserving local similarities, emphasizing neighborhoods and short-range interactions between points.
- The process involves calculating the probability that data point i selects data point j as its neighbor based on their Euclidean distance.
- In the low-dimensional space, a similar probability is calculated using a t-distribution with one degree of freedom. The similarity between the two embedded spaces is measured using the Kullback-Leibler (KL) divergence.

t-distributed Stochastic Neighbor Embedding (t-SNE)

- One notable issue with t-SNE is the crowding problem, where data points are crowded too close together when mapped to a lower-dimensional space. In the original high dimensional space, neighbors may be relatively far apart.
- To address this issue, t-SNE uses a t-distribution instead of a Gaussian distribution, as the former's fatter tails allow for better modeling of distant points and potential outliers.
- The formula for the high dimension space is unchanged.



Dimensionality Reduction Results

- t-SNE visualized distinct groupings of the data. This result demonstrates that t-SNE's ability to preserve local similarities and emphasize the relationship between neighborhoods and short-range interactions between points is most significant when embedding the data into a lower-dimensional space.

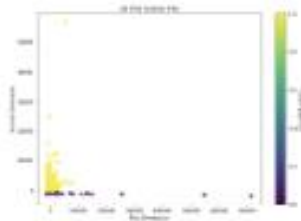


Figure 2: PCA

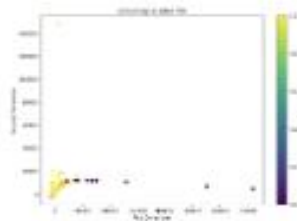


Figure 3: Isomap

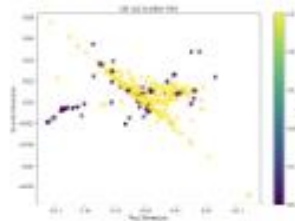


Figure 4: LLE

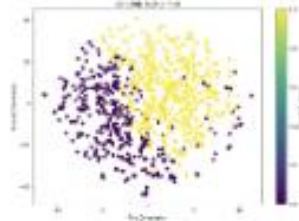


Figure 5: t-SNE

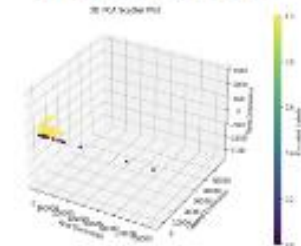


Figure 6: PCA 3D

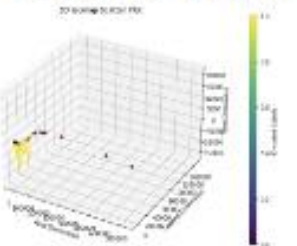


Figure 7: Isomap

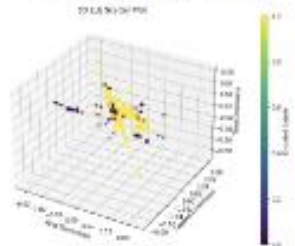


Figure 8: LLE 3D

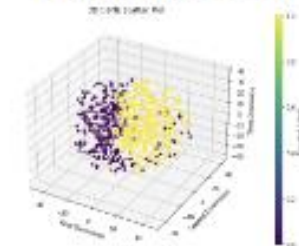
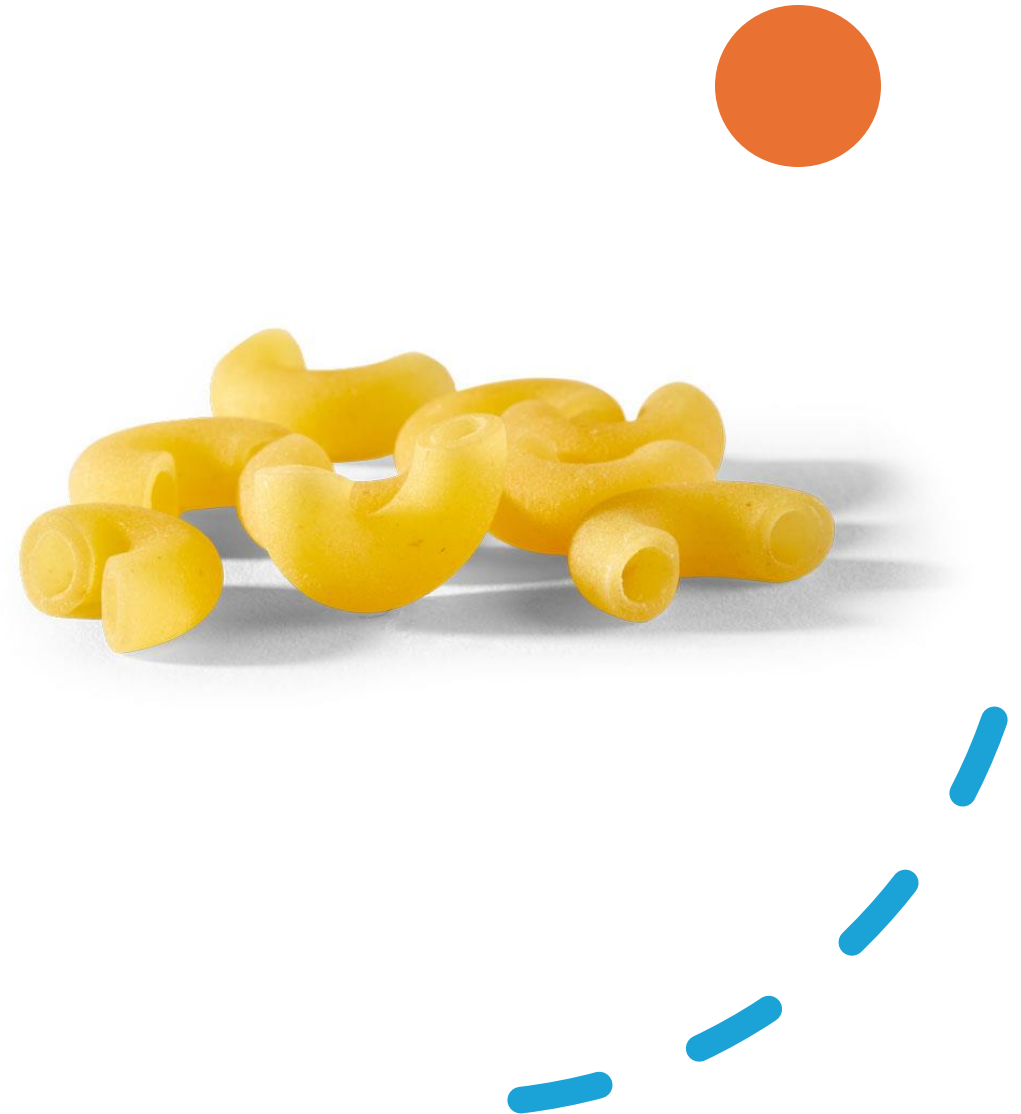


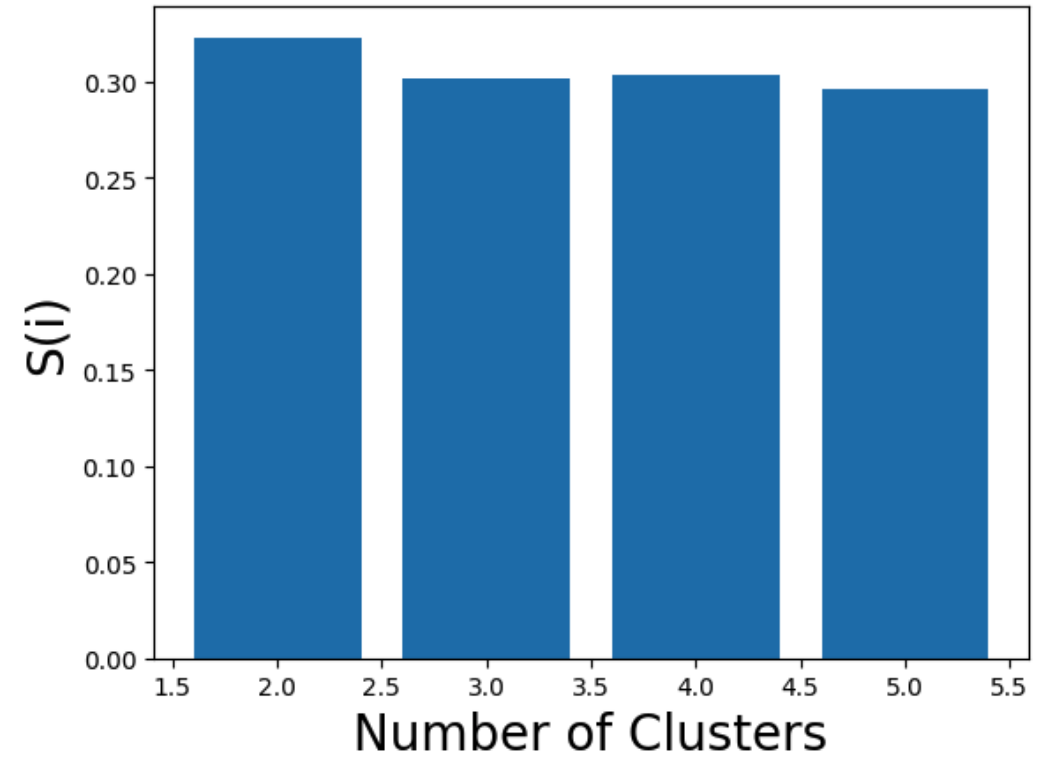
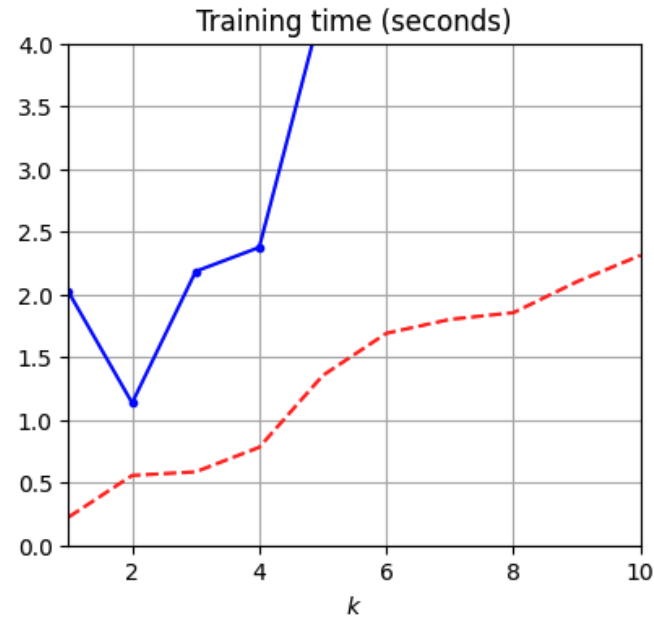
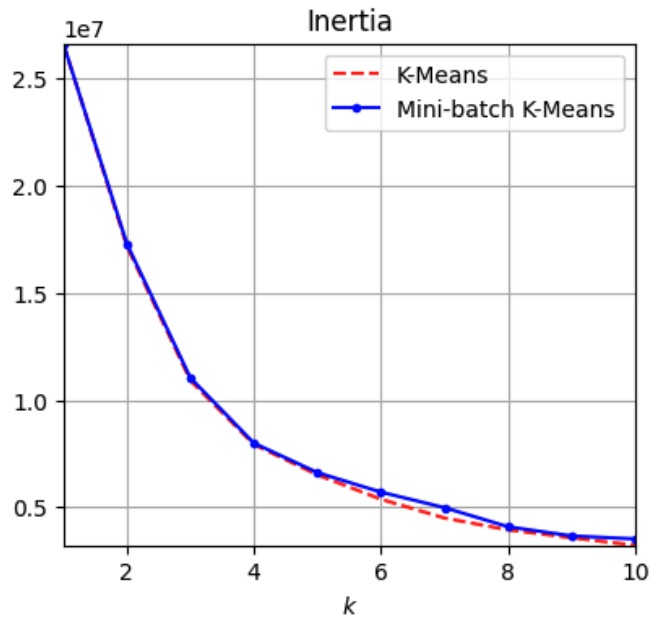
Figure 9: t-SNE 3D

Elbow Method, not Elbow Macaroni!

- **Main Idea**: Use the elbow of the curve in a visualization to help find optimal hyperparameters for our choice of models
- For Agglomerative and K-Means Clustering, let's look at a **Silhouette bar graph** and **Inertia graph** respectively
- $K = 2$ seems the most optimal, that's where the magic happens. Let's use that for our algorithms...
- For epsilon and min sample points, we will be using a semi-supervised approach: **K-Distance Plot**!



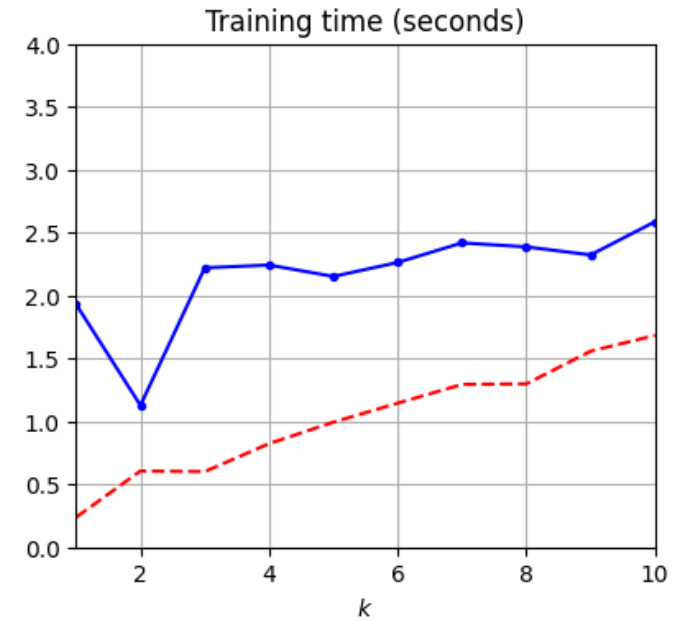
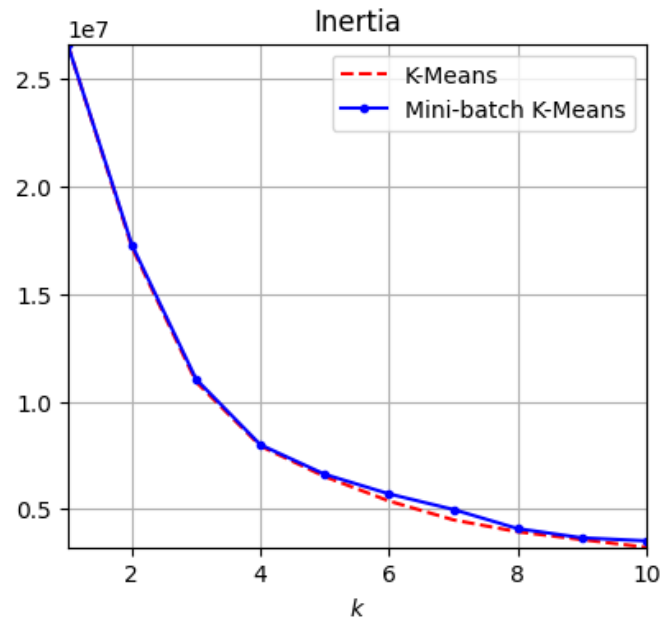
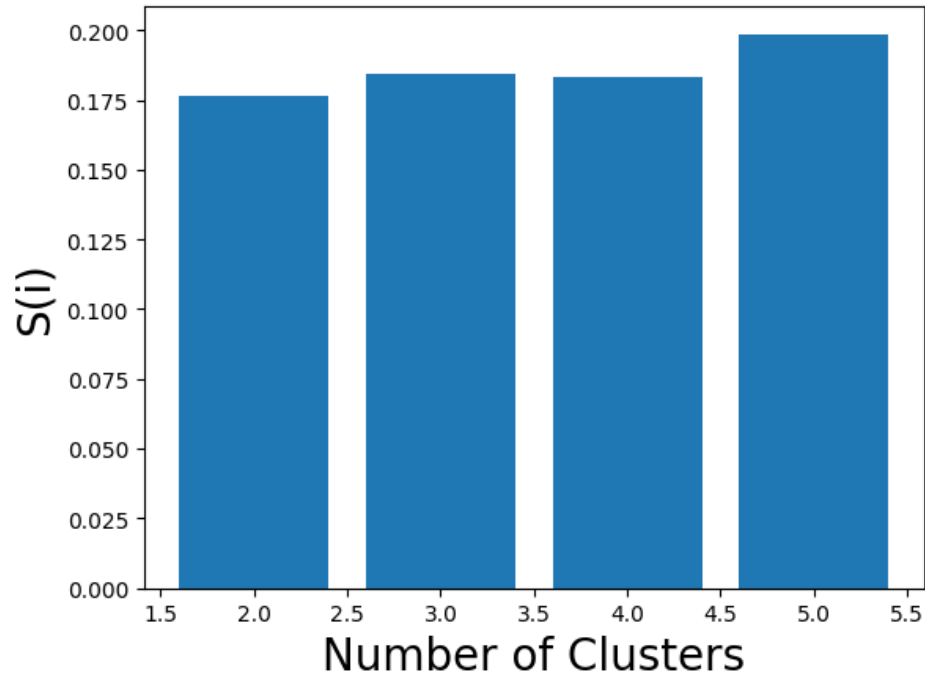
INERTIA/SILHOUETTE METHOD FOR 2D



K=2 looks reasonable
to use

INERTIA/SILHOUETTE METHOD FOR 3D

K=5 doesn't seem
right due to ground
truth knowledge,
but inertia tells us
K=2



Question: How Does K-Distance Work?

- **Main Idea**: Use KNN to determine the epsilon and minpts by graphing the pointwise distances between data and their K-Nearest Neighbors ($K = 2 * \text{dim}$ for large data)
- Sort each of the distances and plot them, then you now have a cubic function looking plot
- Use our friend the handy dandy **Elbow Method**, and this is our epsilon! (theoretically speaking, however once again the hyperparameters here are deterministic/based on domain knowledge)



```
from sklearn.neighbors import NearestNeighbors

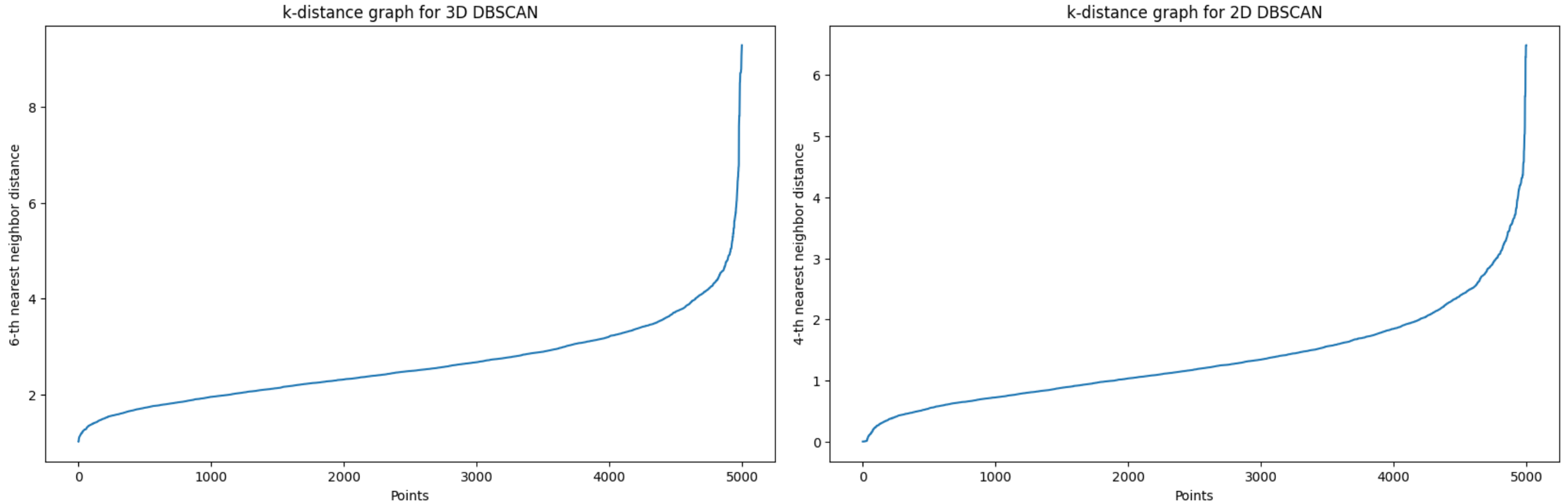
def plot_k_distance_graph(X, k, title):
    neigh = NearestNeighbors(n_neighbors=k)
    neigh.fit(X)
    distances, _ = neigh.kneighbors(X)
    distances = np.sort(distances[:, k-1])
    plt.figure(figsize=(10, 6))
    plt.plot(distances)
    plt.xlabel('Points')
    plt.ylabel(f'{k}-th nearest neighbor distance')
    plt.title(title)
    plt.show()

# Plot k-distance graph
plot_k_distance_graph(tsne_usage2, k=4, title='k-distance graph for 2D DBSCAN')
plot_k_distance_graph(tsne_usage1, k=6, title='k-distance graph for 3D DBSCAN')
```

✓ 0.4s



K-DISTANCE PLOTS FOR 2D and 3D

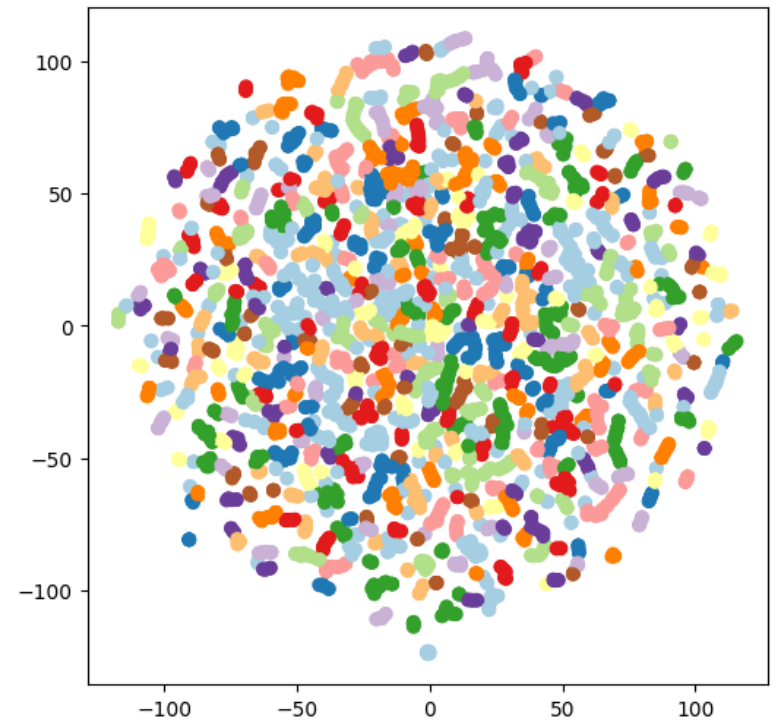
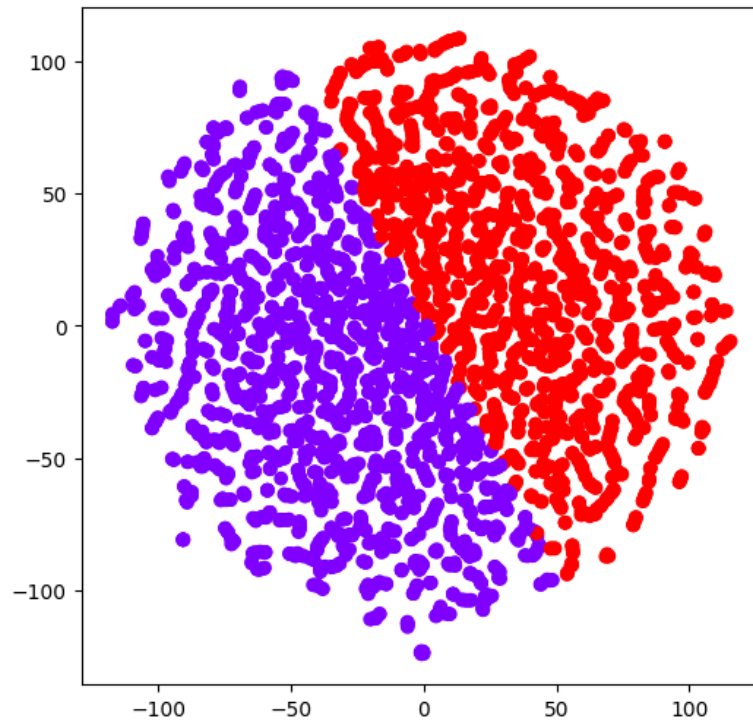
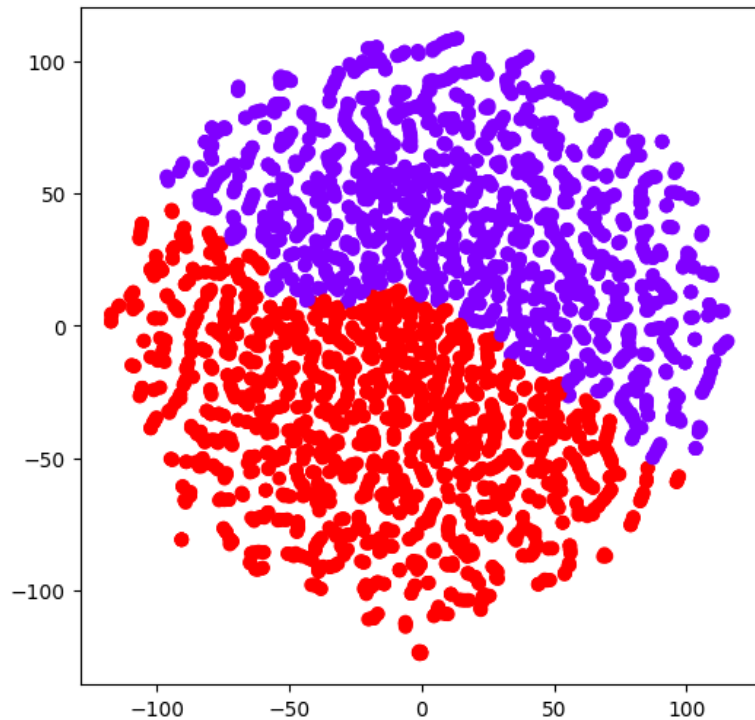


By elbow method, use $\epsilon=2.5$ with $\text{minpts}=2 \times \text{dim} = 2 \times 2 = 4$ for 2D DBSCAN.

Similarly, use $\epsilon=4$ with $\text{minpts} = 2 \times \text{dim} = 2 \times 3 = 6$ for 3D DBSCAN.

Disclaimer: **Not a strict rule, just one of them that we used.** (Don't shoot the messenger please)

Agglomerative Clustering, KMeans, and DBSCAN in 2D



Agglomerative Clustering, KMeans, and DBSCAN in 3D

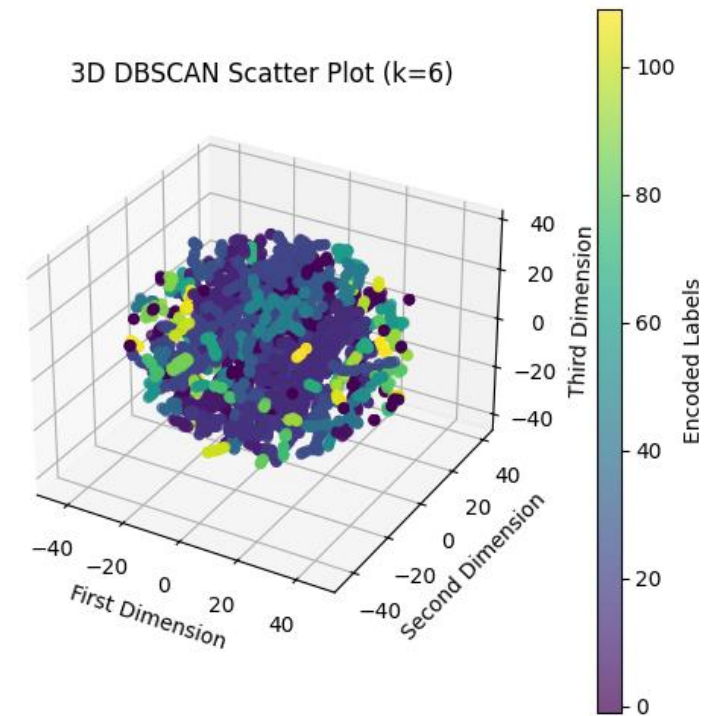
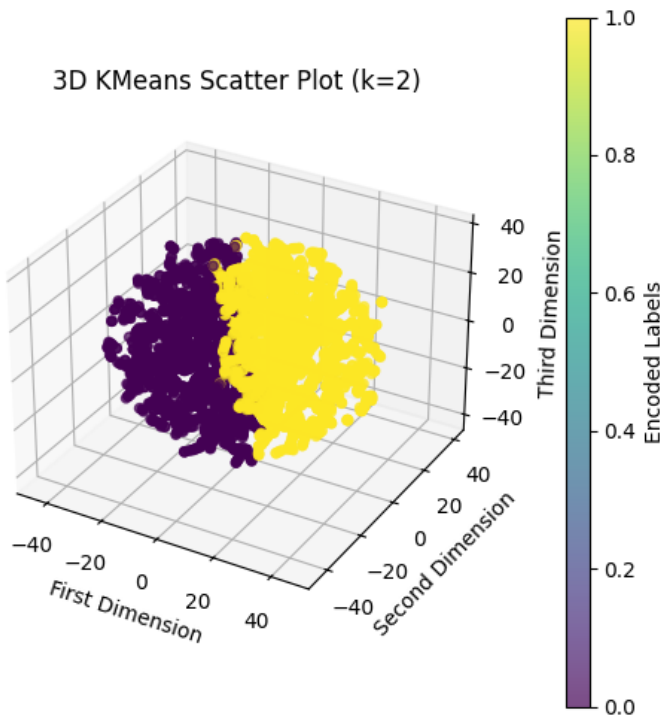
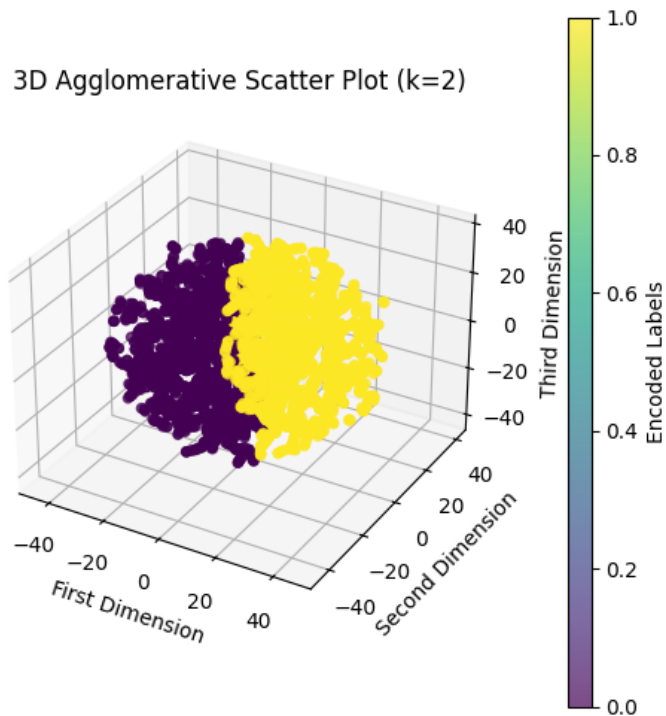


Table 1: Comparison of Clustering Methods in 2 Dimensions

Clustering Method	ARI	Observations
K-Means (Default)	0.2368	Moderate performance.
Mini-Batch K-Means	0.2368	Faster training in batches.
Agglomerative Clustering	0.1557	Handles hierarchical groupings and computationally efficient.
DBSCAN	0.0056	Struggled with sparse clusters, density-based.

Table 2: Comparison of Clustering Methods in 3 Dimensions

Clustering Method	ARI	Observations
K-Means (Default)	0.5346	Best overall performance.
Mini-Batch K-Means	0.5346	Faster training, best performance.
Agglomerative Clustering	0.0155	Performed worse than 2D space.
DBSCAN	0.0159	Struggled with clustering in 3D space.

ANALYSIS / FUTURE WORK / CLOSING REMARKS

- Feature selection by correlation significantly reduced data volume.
- Nonlinear Dimensionality Reduction via T-SNE provided best reduced data while preserving hidden information in both 2D and 3D.
- K-Means performed the best and visually gave the best clustering in 2D and 3D as seen in the Adjusted Rand Index (ARI) table and plots.
- Future Work: Work with updated dataset and use different metrics and hyperparameters for choice of algorithms!

REFERENCES

- [1] University of Florida Information Technology. (2024, February 19). *Spear Phishing on the Rise*. University of Florida. Retrieved from <https://news.it.ufl.edu/security/spear-phishing-on-the-rise/>
- [2] Lee, Seungjoon. STAT 576 Lectures and Code Handouts. CSULB. Accessed Fall 2024.
- [3] DataCamp. A Guide to the DBSCAN Algorithm: Determining the Epsilon Parameter. Accessed 30 November 2024. <https://www.datacamp.com/tutorial/dbscan-clustering-algorithm>
- [4] Prasad, A., & Chandra, S. (2023). PhiUSIIL: A Diverse Security Profile Empowered Phishing URL Detection Framework Based on Similarity Index and Incremental Learning. *Computers & Security*, 103545. doi: <https://doi.org/10.1016/j.cose.2023.103545>



THANKS STAT 576 AND DR. JOON!!

IT'S BEEN A PLEASURE :)

