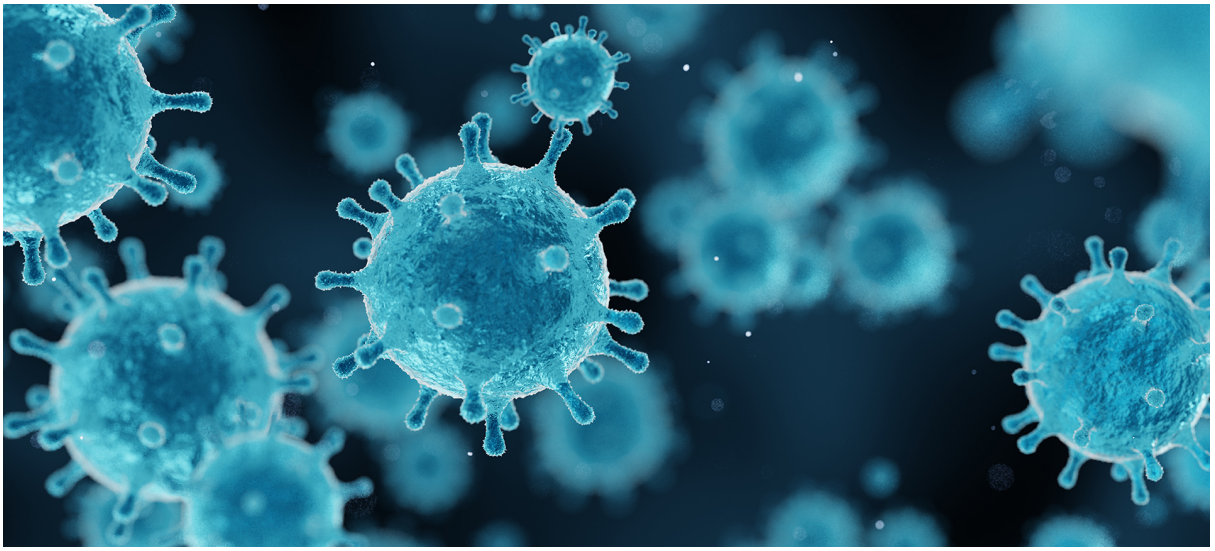


[머신러닝] MACHINE LEARNING PROJECT - PDF (Oh Dong Geun w/ Lee Soo Haeng)

(한글 부제) : 바이러스 vs 인간, 그 단백질의 비교.

1. **Motivation for the Project**

1.1. The announcement of the 2024 Nobel Prize in Chemistry being awarded to AlphaFold inspired this research.

1.2. It sparked the idea of conducting a machine learning project utilizing protein data, given the growing importance of protein structure prediction in scientific advancements.

2. **Reasons for Data Selection and Data Description**

2.1. The Protein Data Bank (PDB) is a globally recognized repository that provides detailed three-dimensional structural data of proteins and other biomolecules. It serves as a vital resource for researchers, offering insights into molecular interactions, protein functions, and biological mechanisms. PDB data

is widely used in structural biology, drug discovery, and machine learning applications.

2.2. We downloaded approximately 1,000 data samples each for humans and viruses from the PDB database to classify and cluster them using machine learning.

2.3. PDB data contains information about a single protein per file, so the features were vectorized to prepare input data for machine learning.

2.4. Below is a description of the features we used.

Feature Name	Description
`A`, `C`, ..., `Y`	Relative frequency of each amino acid type in the protein sequence (normalized counts).
`polar_ratio`	Ratio of polar amino acids (`R, N, D, Q, E, H, K, S, T, Y`) to the total amino acids.
`hydrophobic_ratio`	Ratio of hydrophobic amino acids (`A, C, I, L, M, F, P, W, V`) to the total amino acids.
`isoelectric_point`	The isoelectric point (pI) of the protein sequence, representing the pH at which the protein has no net charge.
`helix_ratio`	Ratio of helical secondary structures (`H`) in the protein to the total secondary structures.
`sheet_ratio`	Ratio of sheet secondary structures (`E`) in the protein to the total secondary structures.
`loop_ratio`	Ratio of loop secondary structures (`C`) in the protein to the total secondary structures.
`chain_length`	Total number of amino acids in the protein chain.
`molecular_weight`	Molecular weight of the protein, calculated as the sum of the weights of all amino acids.
`hydrophilicity_ratio`	Ratio of hydrophilic (polar) amino acids in the protein sequence.

3. **Problem Definition**

3.1 Classification: Human vs. Virus Classification

This section focuses on distinguishing between human and virus proteins using machine learning models. By leveraging labeled data and supervised learning techniques, we aim to accurately classify proteins into their respective categories based on extracted features.

3.2 Clustering

Clustering is used to group proteins based on structural and compositional similarities without relying on predefined labels. This unsupervised learning approach allows us to explore the natural patterns within the data and investigate whether distinct clusters corresponding to human and virus proteins emerge. It helps uncover hidden relationships and validates classification results by identifying meaningful groupings.

4. **Methodology**

4.1. summary:

Data Preparation: Mount Google Drive, load data, handle missing values (replace with mean), and remove outliers (IQR method).

Data Visualization: Plot key variable distributions and correlation heatmap.

Data Scaling: Separate features and targets, then apply standard scaling.

Modeling: Split data into train/test sets, define models, tune hyperparameters (nested cross-validation), and evaluate test performance.

Ensemble Models: Add VotingClassifier and StackingClassifier for improved performance.

Result Summary: Organize results in a DataFrame and visualize performance metrics.

Decision Boundaries: Visualize decision boundaries using the two most important features.

Feature importance: Random Foreset Feature Importance, Logistic Regression Coeficients, Permutation Importance

Clustering (Unsupervised Learning):

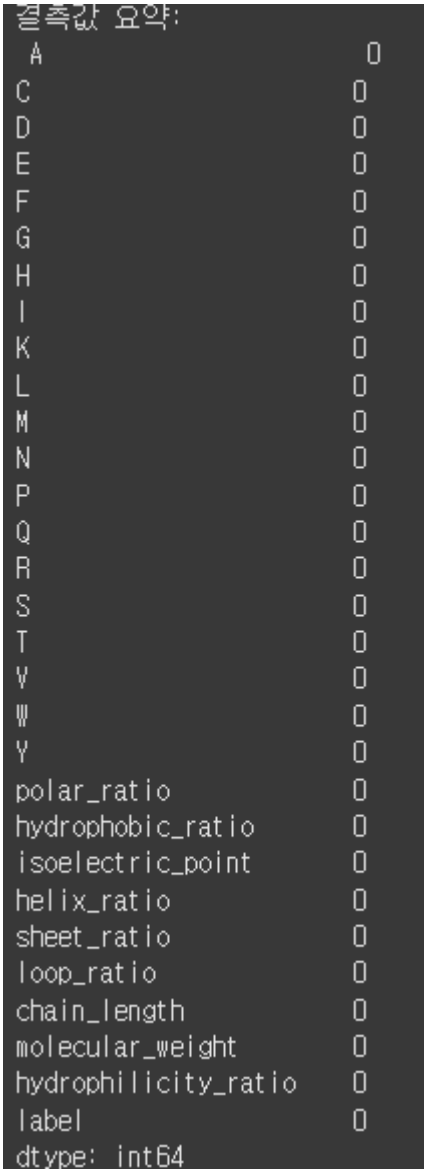
Select key features (Gradient Boosting, PCA, LDA).

Build and evaluate clustering models.

Optimize clustering using silhouette coefficients (KMeans).

4.2. Preprocessing

4.2.1. Handling Missing Values

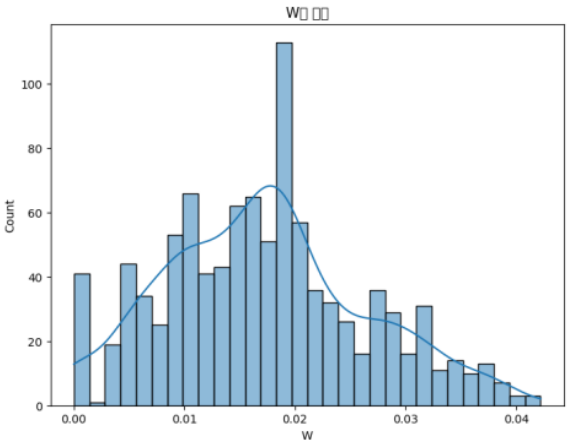


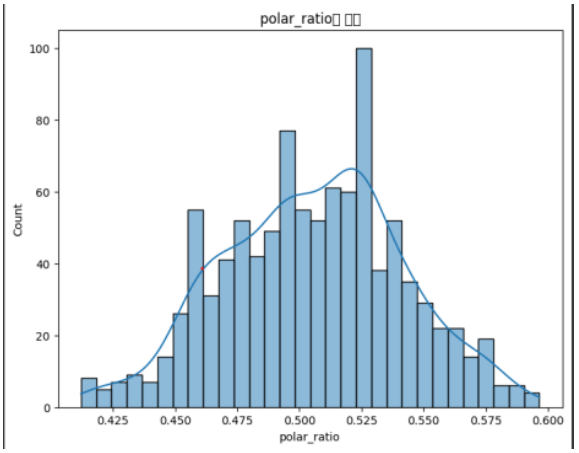
4.2.2. Outlier Handling

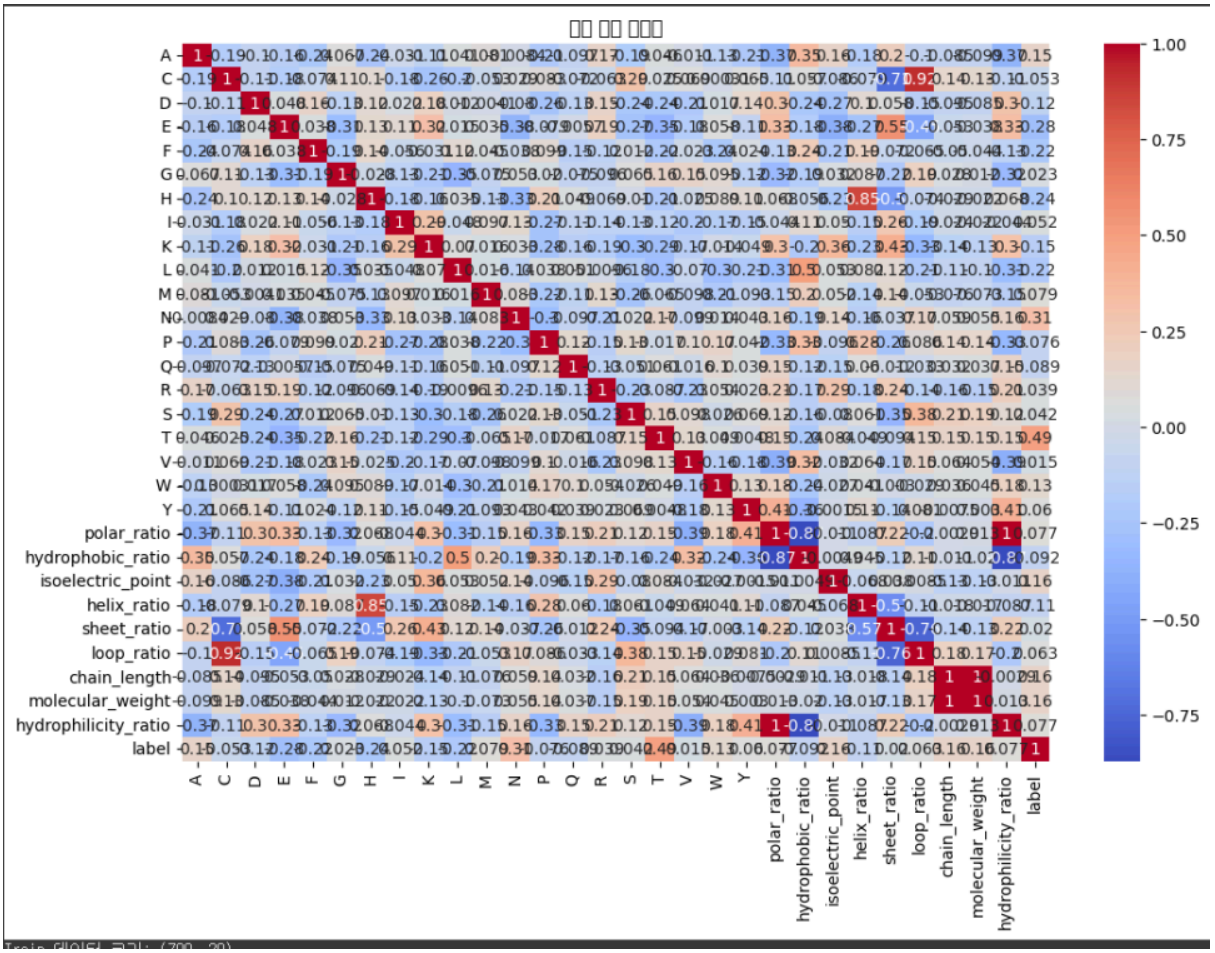
```python
# Step 3: 이상값 처리
# IQR 기준 이상값 제거 함수 정의
def remove_outliers_iqr(df, columns):
    for col in columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
    return df

# 이상값 제거
data = remove_outliers_iqr(data, data.select_dtypes(include=['float64', 'int64']).columns)
```

4.2.3. Data Visualization



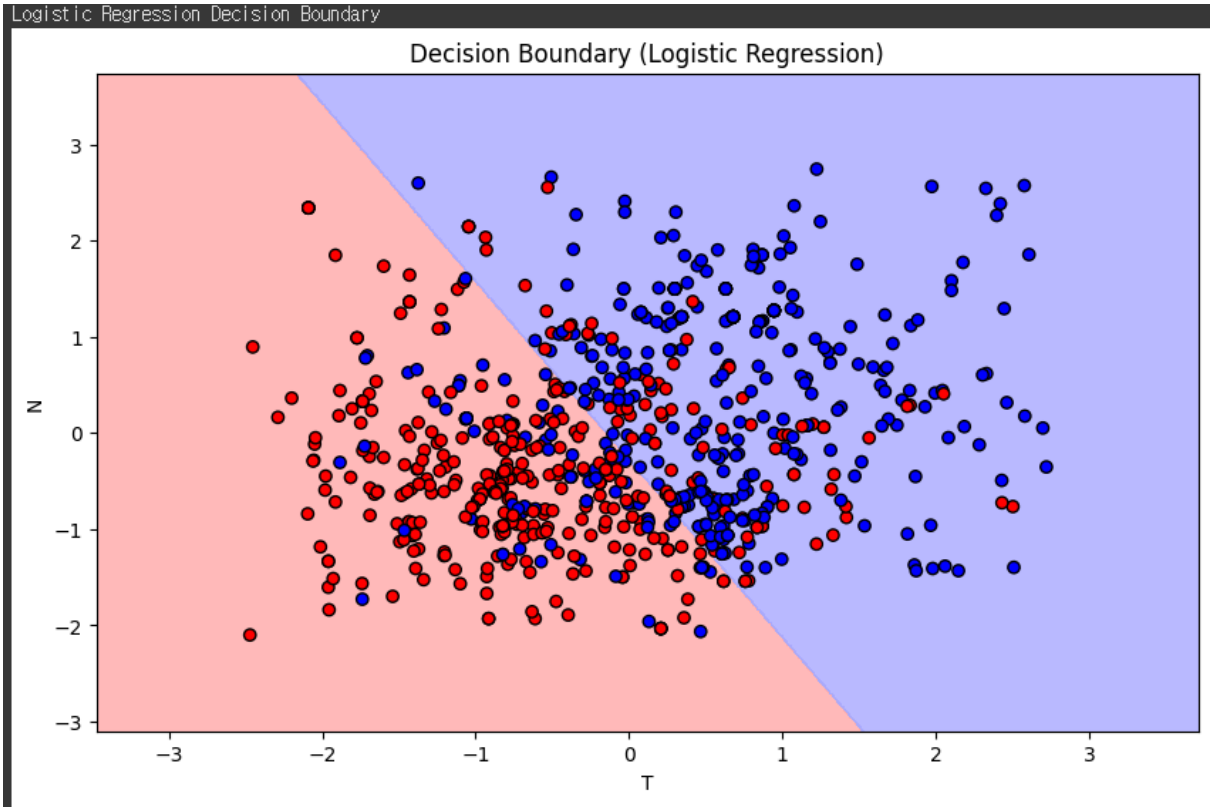




4.3. Classification

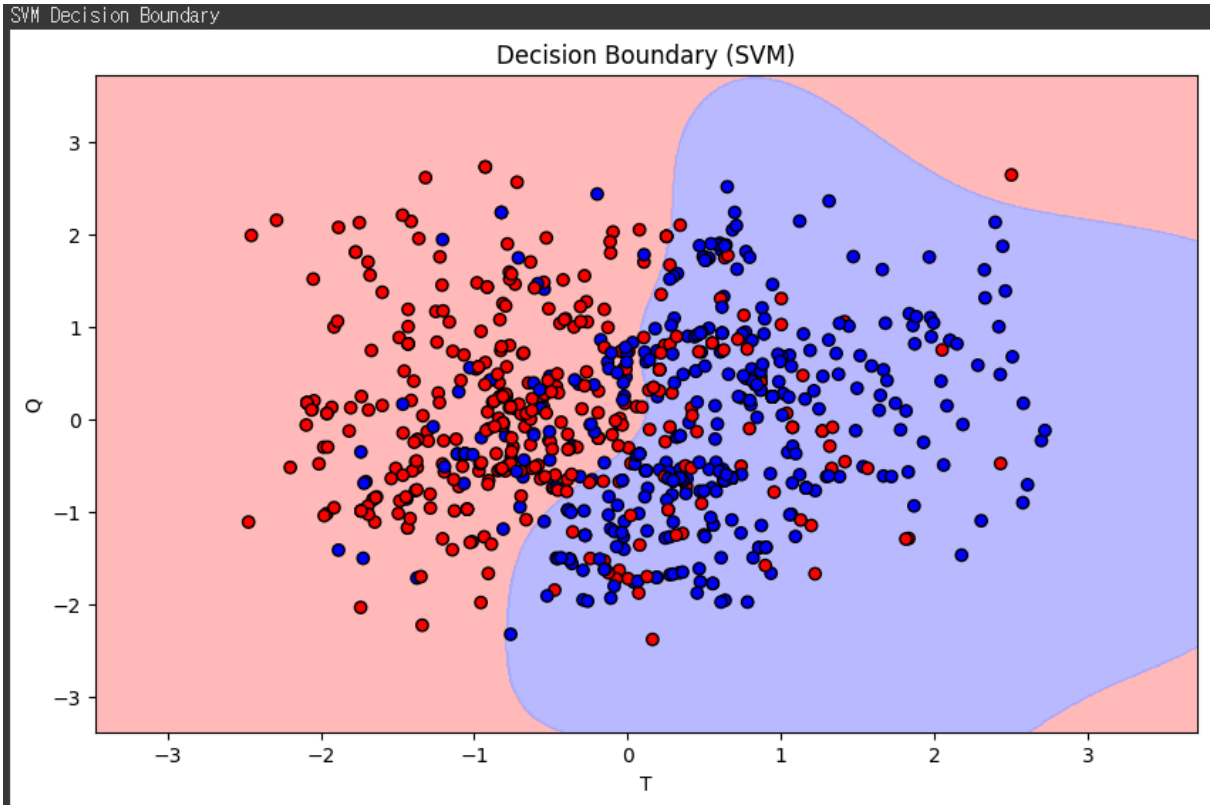
4.3.1. Logistic Regression

```
Logistic Regression Hyperparameter Tuning and Cross-Validation
Best Parameters: {'C': 1}
Cross-Validation Accuracy: 0.8007120253164557
Test Accuracy: 0.725
Test F1 Score: 0.7533632286995515
```



4.3.2. SVM

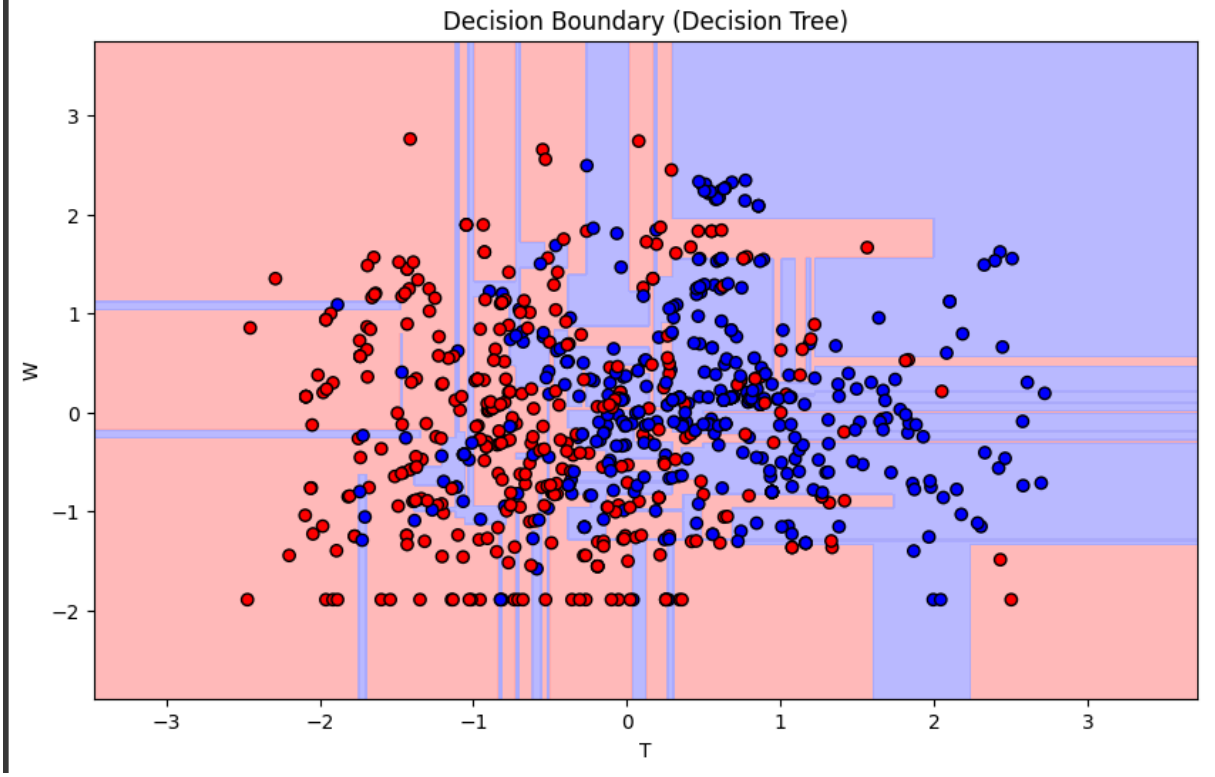
```
SVM Hyperparameter Tuning and Cross-Validation
Best Parameters: {'C': 10, 'kernel': 'rbf'}
Cross-Validation Accuracy: 0.8796677215189874
Test Accuracy: 0.88
Test F1 Score: 0.8888888888888888
```



4.3.3. Decision Tree

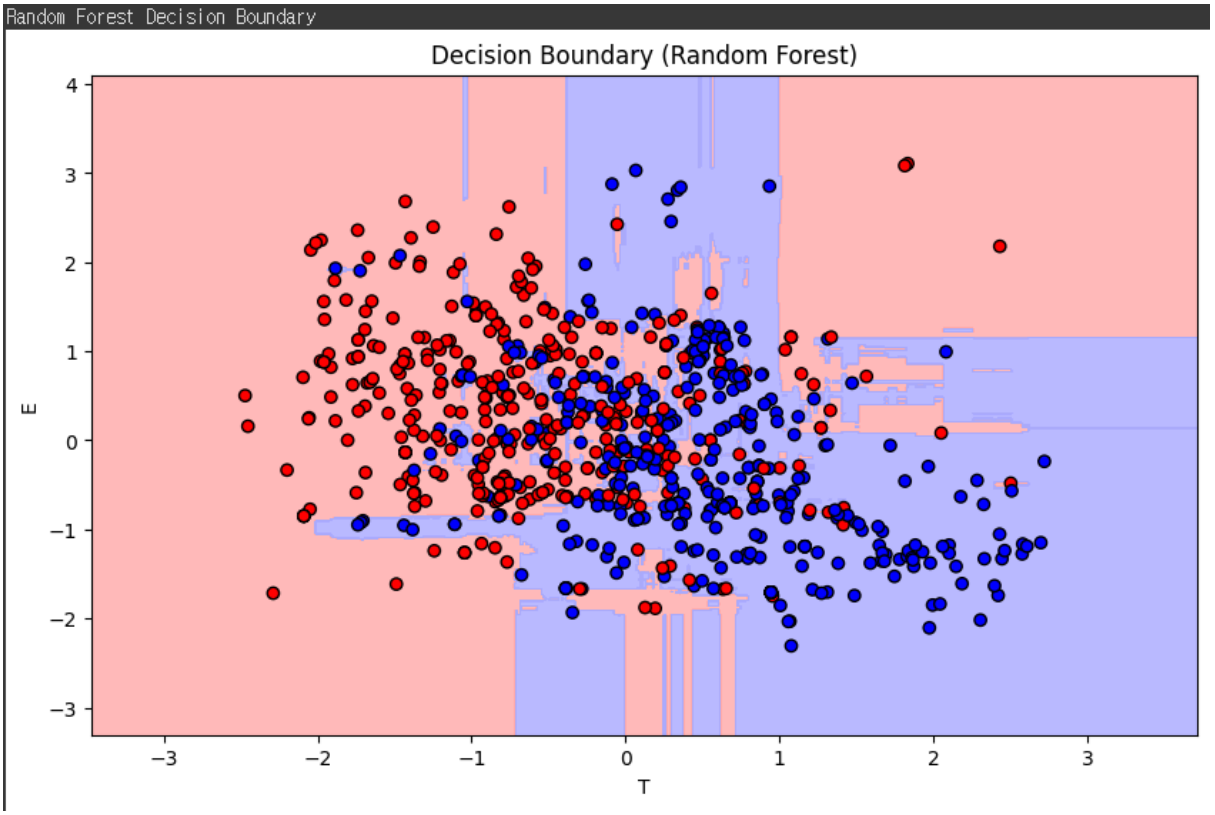
```
Decision Tree Hyperparameter Tuning and Cross-Validation
Best Parameters: {'max_depth': None, 'min_samples_split': 10}
Cross-Validation Accuracy: 0.8270253164556962
Test Accuracy: 0.795
Test F1 Score: 0.7980295566502463
```

Decision Tree Decision Boundary



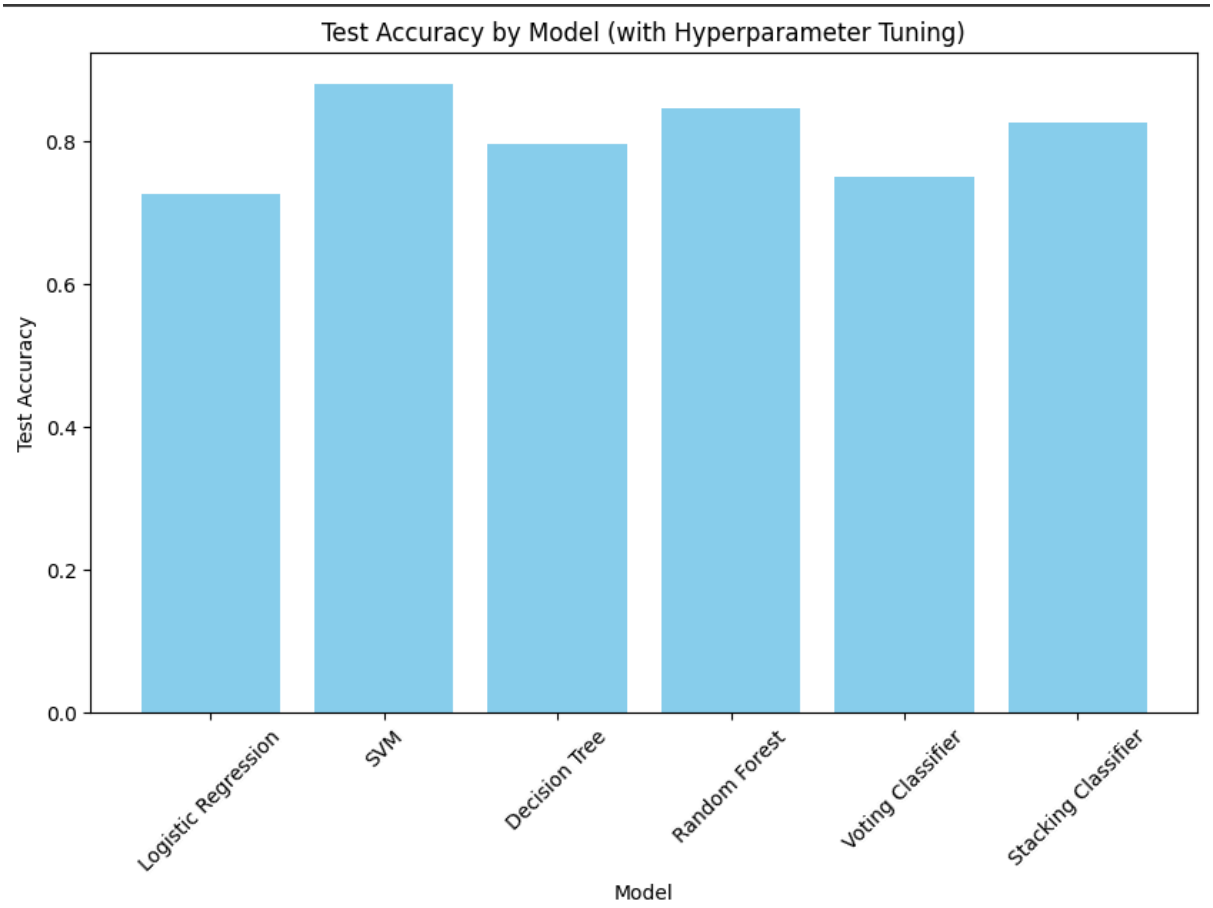
4.3.4 Random Forest

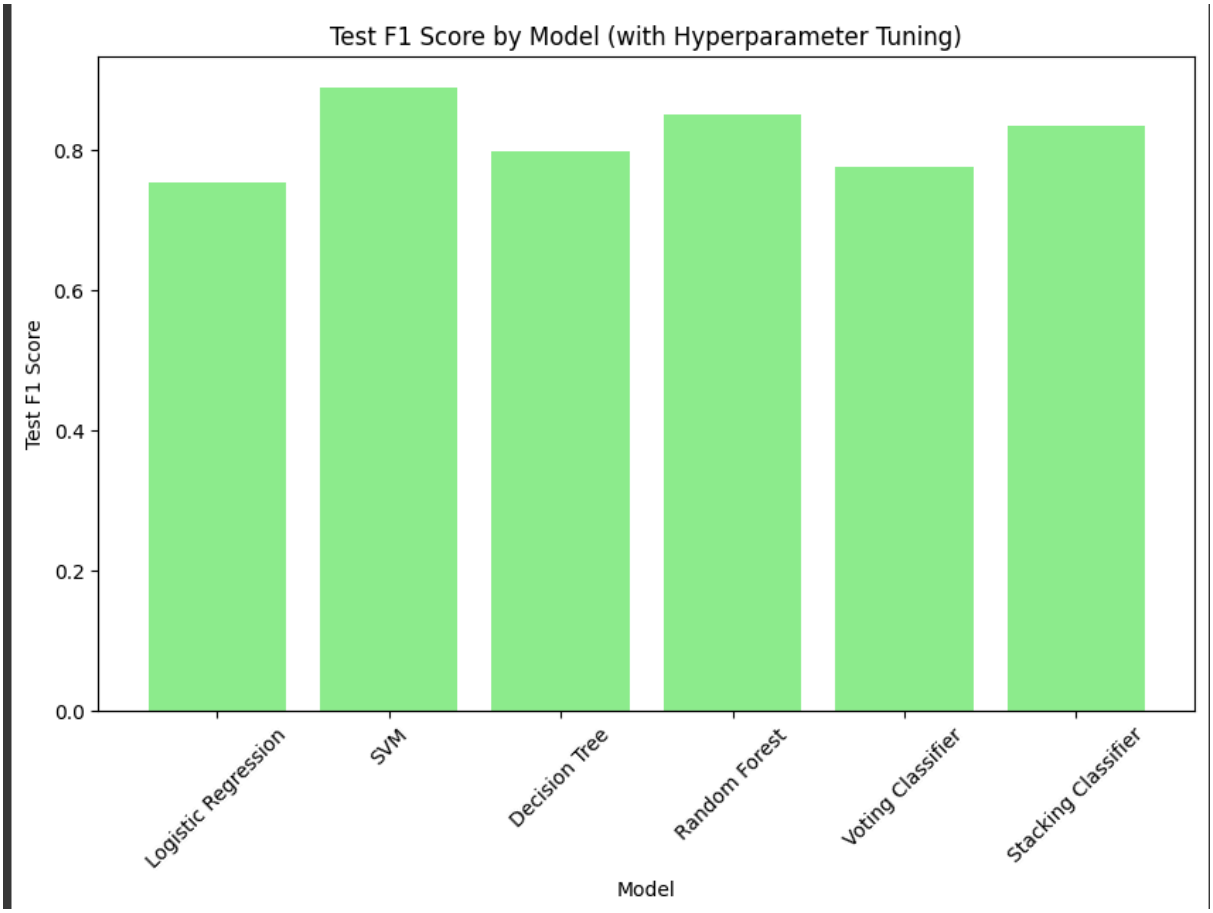
```
Random Forest Hyperparameter Tuning and Cross-Validation
Best Parameters: {'max_depth': None, 'n_estimators': 200}
Cross-Validation Accuracy: 0.8846835443037975
Test Accuracy: 0.845
Test F1 Score: 0.8502415458937198
```



4.3.5 Hyperparameter Grid Search

```
Model Performance with Hyperparameter Tuning:
                 Model                                  Best Parameters
0  Logistic Regression                                          {'C': 1}
1                  SVM                      {'C': 10, 'kernel': 'rbf'}
2        Decision Tree  {'max_depth': None, 'min_samples_split': 10}
3        Random Forest      {'max_depth': None, 'n_estimators': 200}
4    Voting Classifier                                            None
5  Stacking Classifier                                            None
```


Test Accuracy by Model (with Hyperparameter Tuning)
Test Accuracy
0.0
0.2
0.4
0.6
0.8
Logistic Regression
SVM
Decision Tree
Random Forest
Voting Classifier
Stacking Classifier
Model

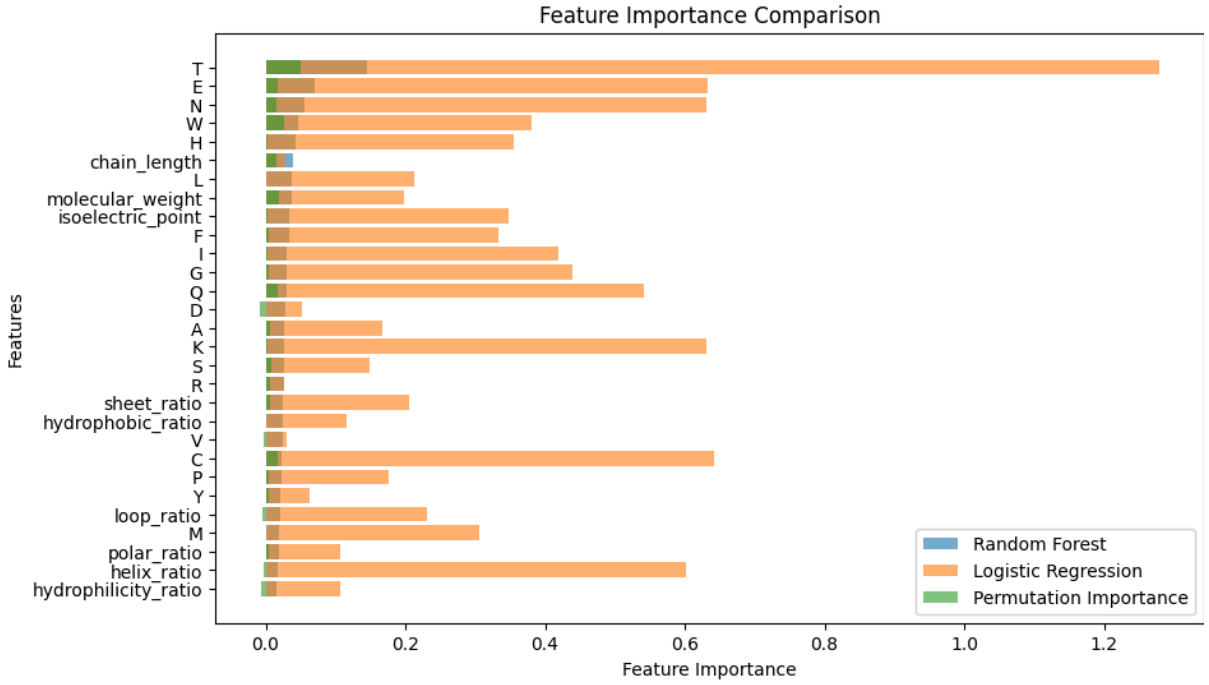

Test F1 Score by Model (with Hyperparameter Tuning)
Test F1 Score
0.0
0.2
0.4
0.6
0.8
Logistic Regression
SVM
Decision Tree
Random Forest
Voting Classifier
Stacking Classifier
Model

4.3.6 Ensemble Learning Experiments

```
Voting Classifier Training and Evaluation
Test Accuracy: 0.75
Test F1 Score: 0.7747747747747747

Stacking Classifier Training and Evaluation
Test Accuracy: 0.825
Test F1 Score: 0.8341232227488151
```

4.3.7 Feature importance

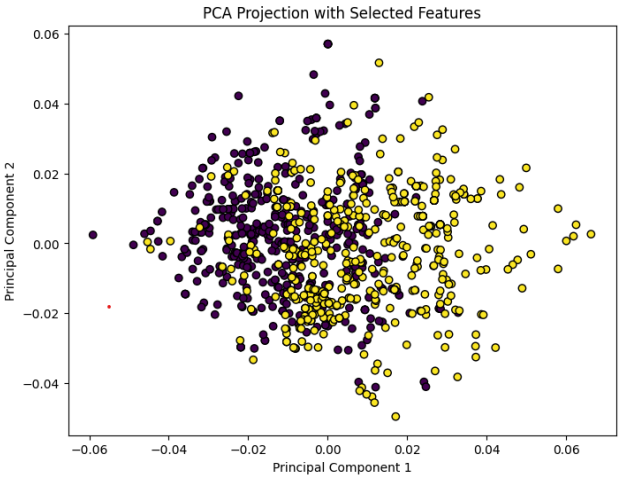


4.4. Clustering

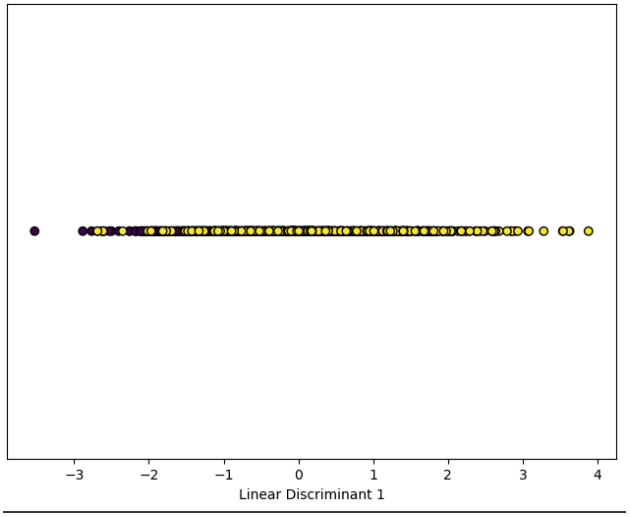
4.4.1 Key Feature Selection Using Gradient Boosting

```
Selected Features (Gradient Boosting): Index(['N', 'T'], dtype='object')
```

4.4.2 PCA

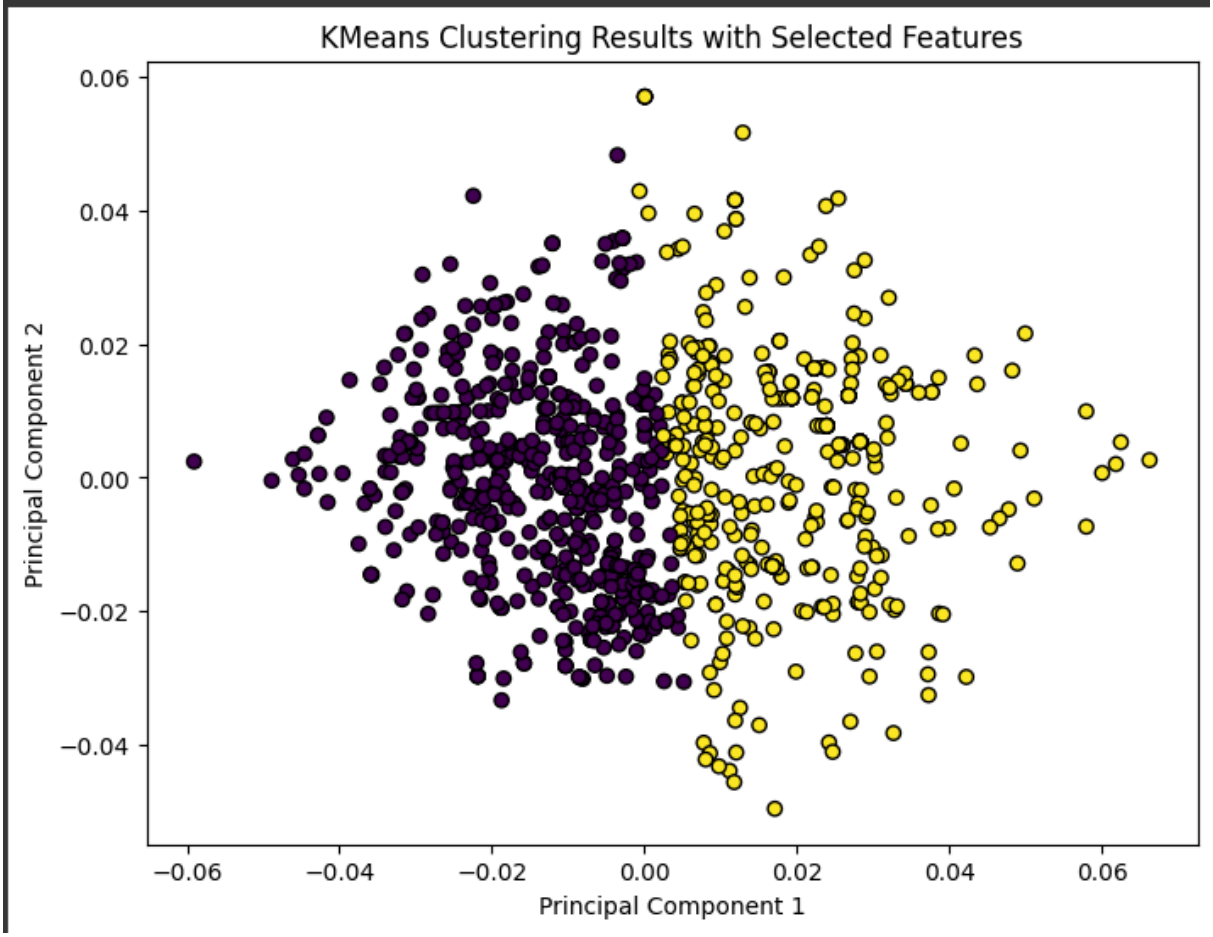


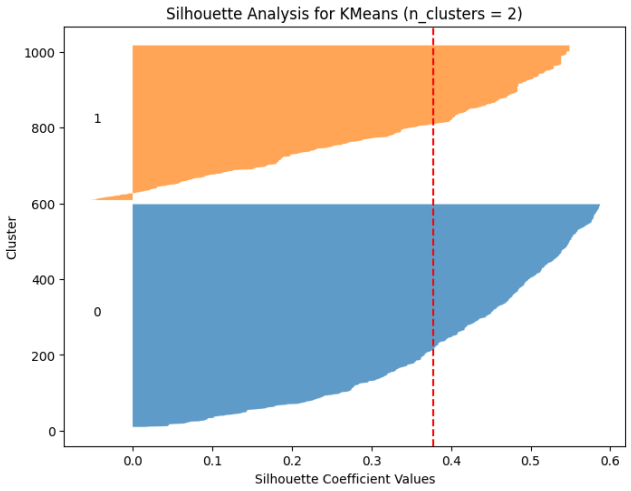
4.4.3 LDA

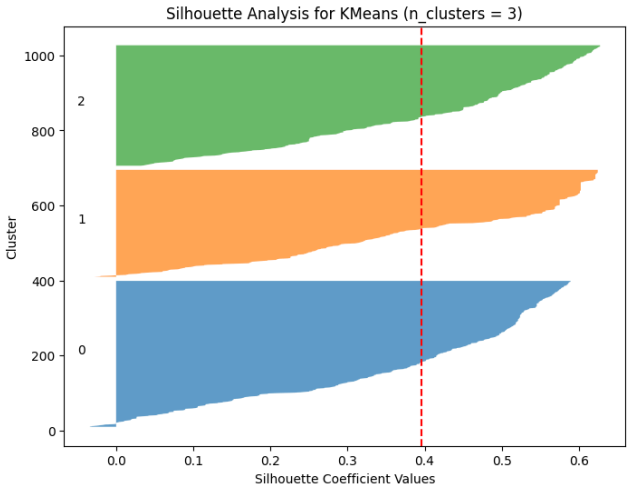


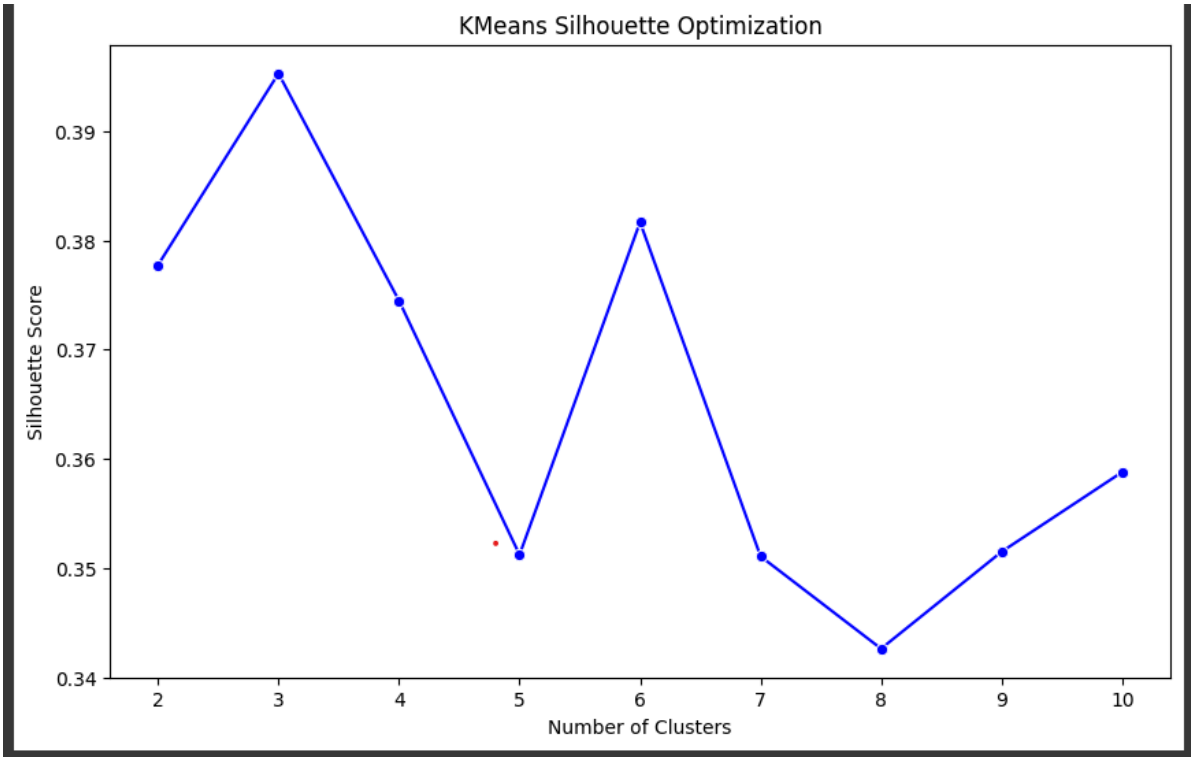
4.4.4. KMEANS

```
KMeans Clustering:
Silhouette Score: 0.3777
Davies-Bouldin Index: 1.0786
```




Silhouette Analysis for KMeans (n_clusters = 2)
Cluster
Silhouette Coefficient Values
1
0
0
200
400
600
800
1000
0.0
0.1
0.2
0.3
0.4
0.5
0.6


Silhouette Analysis for KMeans (n_clusters = 3)
Cluster
Silhouette Coefficient Values
2
1
0
0
200
400
600
800
1000
0.0
0.1
0.2
0.3
0.4
0.5
0.6

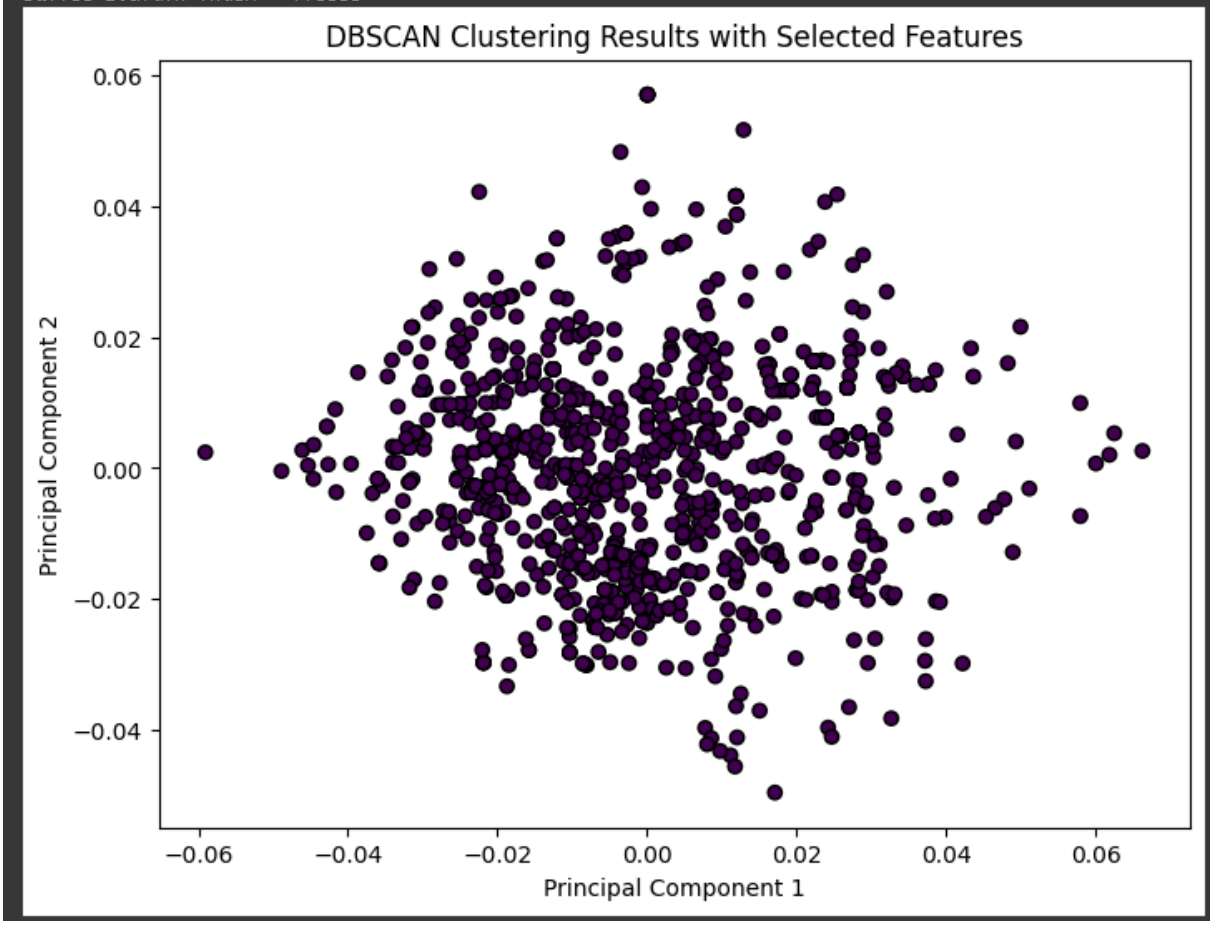



```
Optimal Number of Clusters: 3

Silhouette Optimization Results:
   Clusters  Silhouette Score
0         2          0.377743
1         3          0.395280
2         4          0.374481
3         5          0.351228
4         6          0.381664
5         7          0.351081
6         8          0.342630
7         9          0.351502
8        10          0.358795
```

4.4.5 DBSCAN

```
DBSCAN Clustering:
Silhouette Score: -1.0000
Davies-Bouldin Index: -1.0000
```



5. **Results**

5.1 Classification

It was identified that the amino acid Threonine (T) plays a decisive role in distinguishing human proteins from viral proteins.

(Based on domain knowledge research, the following insight was obtained: Threonine acts as a crucial biochemical marker for distinguishing human and viral proteins. Its phosphorylation patterns, polarity characteristics, distribution in amino acid sequences, and structural contributions serve as key elements reflecting functional and structural differences between the two protein groups. These features aid in understanding protein interactions and infection

mechanisms and hold potential for quantitative analysis to classify viral proteins effectively.)

A classification model was trained that achieved a reasonably significant F1 score for distinguishing human and viral proteins.

The model showed the best performance (F1 score: 0.89) when using SVM with hyperparameters C (Regularization Parameter): 10 and Kernel: RBF (Radial Basis Function Kernel).

5.2 Clustering

The distribution after PCA transformation was similar to the distribution obtained using KMeans clustering.

While the silhouette score was highest when the number of clusters was 3, the data was best explained with 2 clusters.

DBSCAN did not produce meaningful results.

6. **Importance and Applications**

The amino acid Threonine (T) acts as a significant biochemical marker for distinguishing human proteins from viral proteins. Threonine's phosphorylation patterns, polarity characteristics, distribution within amino acid sequences, and structural contributions serve as key factors reflecting functional and structural differences between the two groups. These properties aid in understanding protein interactions and infection mechanisms while enhancing the performance of quantitative analyses and classification models for viral proteins. This study achieved an F1 score of 0.89 using a machine learning model that leveraged these characteristics, demonstrating that features derived from Threonine and protein sequences effectively provide learnable patterns that capture distinctions between human and viral proteins. The integration of domain-specific biochemical knowledge with machine learning techniques highlights a promising approach to addressing protein classification problems.

To further enhance the applicability of protein classification models utilizing Threonine, two advanced approaches are proposed: deep learning techniques and novel vectorization methods. First, a grid parsing (Grid Parsing) method is proposed for vectorizing 3D structural data. This involves dividing the 3D protein structure into grids and quantifying the physical/chemical properties

(e.g., atom types, polarity, charge distribution) within each grid to create feature vectors. This approach effectively preserves local structural characteristics, enabling the representation of differences between human and viral proteins. Second, deep learning techniques are recommended to learn from structural and sequential protein data. Models such as 3D CNNs can process 3D protein structures directly, while RNNs and Transformer-based models can capture temporal dependencies in protein sequence data. These deep learning-based approaches can learn nonlinear and complex data patterns, potentially surpassing traditional machine learning models. The combination of grid parsing and deep learning maximizes data representation capacity, offering enhanced precision and predictive power.