# MACHINE LEARNING HOMEWORK

Train DataSet: 10422 X 70
Test DataSet: 2606 X 69 (AGA frequency missing)

**Kingdom classes** (11): arc (archaea), bct (bacteria), phg (bacteriophage), plm (plasmid), pln (plant), inv (invertebrate), vrt (vertebrate), mam (mammal), rod (rodent), pri (primate), vrl (virus).
**Predictive features**: relative frequencies of 64 possible codons (+ DNA type → we may choose to include it or not maybe based on performance assessing)

## 1. Preliminary Data Analysis

*Perform a preliminary analysis on the data. For instance, but not limited to, visualize samples, identify if features (i.e., codon frequencies) are correlated, determine which are most correlated with the target classes, and inspect the distribution of samples among classes.*

Data Visualization and Correlation

- **Sample Visualization**: **PCA** (Principal Component Analysis), **t-SNE** (t-Distributed Stochastic Neighbor Embedding) or **uMAP** to reduce dimensionality and visualize the data in 2D or 3D.
- **Feature Correlation**: Calculate the **correlation matrix** between codon frequencies and represent it using a **heatmap** to identify highly correlated features.
- **Sample Distribution**: **barplot** or **histograms** to visualize the distribution of samples across different classes for both **DNA type** and **Kingdom**.

*Using clustering, study if there are structures in the data that allow samples from different classes (both DNA type and Kingdom) to be easily identified. Compare the performance of different clustering algorithms and distance measures using the metrics presented during the course.*

Clustering

- **Clustering Algorithms**: Apply different clustering algorithms such as **K-means**, **DBSCAN**, **K-medoids (PAM)** and **Agglomerative Clustering** to identify structures in the data.
- Compare performances of different distance measures and linkage methods (see lab.06)
- **Evaluation Metrics**: Use metrics like **silhouette score**, *Davies-Bouldin index, or adjusted Rand index* to compare the performance of the various algorithms. Use also Precision, Recall and Entropy indexes (external indexes).
- **Cluster Visualization**: Use **dimensionality reduction (PCA and TSNE)** techniques to visualize the clusters and evaluate class separability.

## 2. Classification of Samples into Kingdoms

*Perform classification to classify organisms into the 11 Kingdom classes. Perform features selection, compare different algorithms and identify the best. Finally, test the performance of the best algorithm on the test set.*

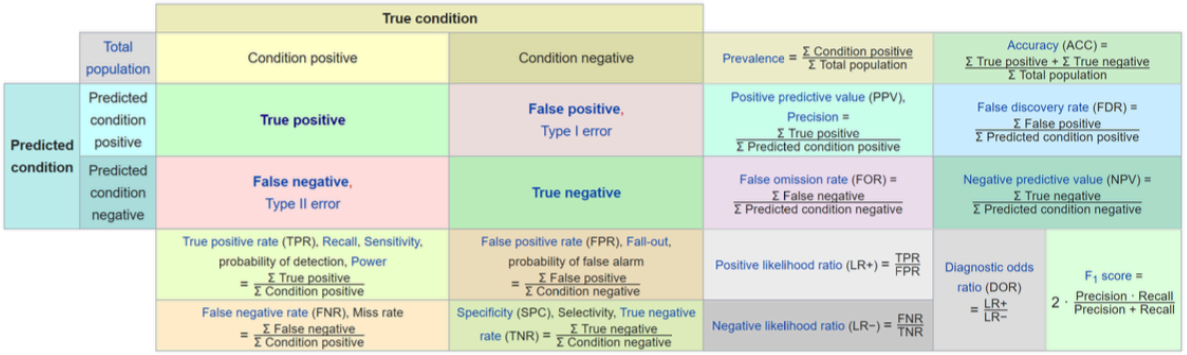
Different types of classification on all features, then feature selection and then classification on selected features. Finally classification comparison.

Feature Selection

- **Feature Selection Methods**: Use feature selection techniques: **Best Subset Selection** (impossible but comment about it) **Forward/Backward feature selection** and **Lasso** (select best tuning parameter value via GridSearchCV). For each classifier identify the best version and compare them with each other (k-fold cross-validation and test set validation)
- *Other possible techniques (not seen during lessons): Random Forest feature importance, Recursive Feature Elimination (RFE), or chi-square-based methods*.

Algorithm Comparison

- **Classification Algorithms**: Logistic Regression, LDA (Linear Discriminant Analysis), QDA (Quadratic Discriminant Analysis), K-NN (select k via cross-validation), SVM (select C via GridSearchCV) with different kernel functions (dth-Degree Polynomial (GridSearchCV), Radial basis, Neural Network), Random Forest, *Naive Bayes, Gradient Boosting* + applied PCA (dimensionality reduction) ?
- **Cross-Validation**: Use cross-validation techniques (e.g., k-fold cross-validation) to evaluate algorithm performance.
- **Performance Metrics**: Compare the performance of algorithms using metrics such as accuracy, precision, recall, F1-score, and AUC-ROC.




Test Set Performance

- **Apply Best Model**: Apply the best classification model to the test set and evaluate its performance using the same metrics.

### 3. Recovery of AGA Codon Frequency in Test Samples

*Train a regressor to predict the value of the AGA feature given the remaining ones → compare different regression algorithms through robust evaluation techniques. Since AGA features are missing in test samples, use only training data for this step.*

Training a Regressor

- **Regression Algorithms**: Use various regression algorithms like Linear Regression, Ridge, Lasso, KNN Regressor, Random Forest Regressor, *and Gradient Boosting Regressor.*
- **Performance Evaluation**: Compare the performance of regressors using metrics like Mean Squared Error (MSE), $R^2$ score.
- Regression results are silly but we can comment on how many features we need to obtain correct results and compare our results with the one we expect (Which are the most useful information that the model is using to predict the missing frequencies?). Is collinearity present in our setting? How is it affecting our result?

### 4. Using the Regression Model to Predict AGA Codon Frequency and Improve Classification

*Use the regression model to recover the AGA codon frequency by predicting its value in each test sample. Determine if the test performance of the best classification model found at step (2.) improves if the AGA frequency values are also used for prediction.*

Predicting AGA Frequency

- **Prediction**: Use the best regression model to predict the missing AGA values in test samples. Evaluating Impact on Classification Model
- **Re-training the Classification Model**: Re-train the best classification model using the training (?) dataset with AGA values.
- **Performance Evaluation**: Compare the performance of this new model with the original model to see if accuracy has improved.

**Implementation in Python**

Here's an example of Python code to get started:

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
from sklearn.metrics import silhouette_score
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, cross_val_score
```

```
from sklearn.metrics import classification_report, mean_squared_error, r2_score

# Load data
train_df = pd.read_csv('train.csv')
test_df = pd.read_csv('test.csv')

# Preprocessing and visualization
# Calculate correlation matrix
corr_matrix = train_df.iloc[:, 3:].corr()
plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, annot=True, fmt='.2f')
plt.title('Heatmap of Codon Frequency Correlations')
plt.show()

# PCA for visualization
scaler = StandardScaler()
scaled_features = scaler.fit_transform(train_df.iloc[:, 3:])
pca = PCA(n_components=2)
pca_result = pca.fit_transform(scaled_features)
plt.scatter(pca_result[:, 0], pca_result[:, 1], c=train_df['Kingdom'])
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.title('PCA of Codon Frequencies')
plt.show()

# Clustering
kmeans = KMeans(n_clusters=11)
kmeans_labels = kmeans.fit_predict(scaled_features)
print(f'Silhouette Score for KMeans: {silhouette_score(scaled_features, kmeans_labels)}')

# Classification
X = train_df.iloc[:, 3:]
y = train_df['Kingdom']
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

# Random Forest Classifier
rf_clf = RandomForestClassifier()
rf_clf.fit(X_train, y_train)
y_pred = rf_clf.predict(X_val)
print(classification_report(y_val, y_pred))

# Regression for AGA codon
X_train_reg = train_df.iloc[:, 3:].drop('AGA', axis=1)
y_train_reg = train_df['AGA']
rf_reg = RandomForestRegressor()
rf_reg.fit(X_train_reg, y_train_reg)

# Predicting missing AGA values
X_test = test_df.iloc[:, 2:]
aga_pred = rf_reg.predict(X_test)
test_df['AGA'] = aga_pred

# Re-training the classification model with predicted AGA values
X_test_full = test_df.iloc[:, 2:]
final_predictions = rf_clf.predict(X_test_full)
```

This is just a starting point for your analysis. Be sure to further explore the data and refine the clustering, classification, and regression algorithms to achieve the best results.