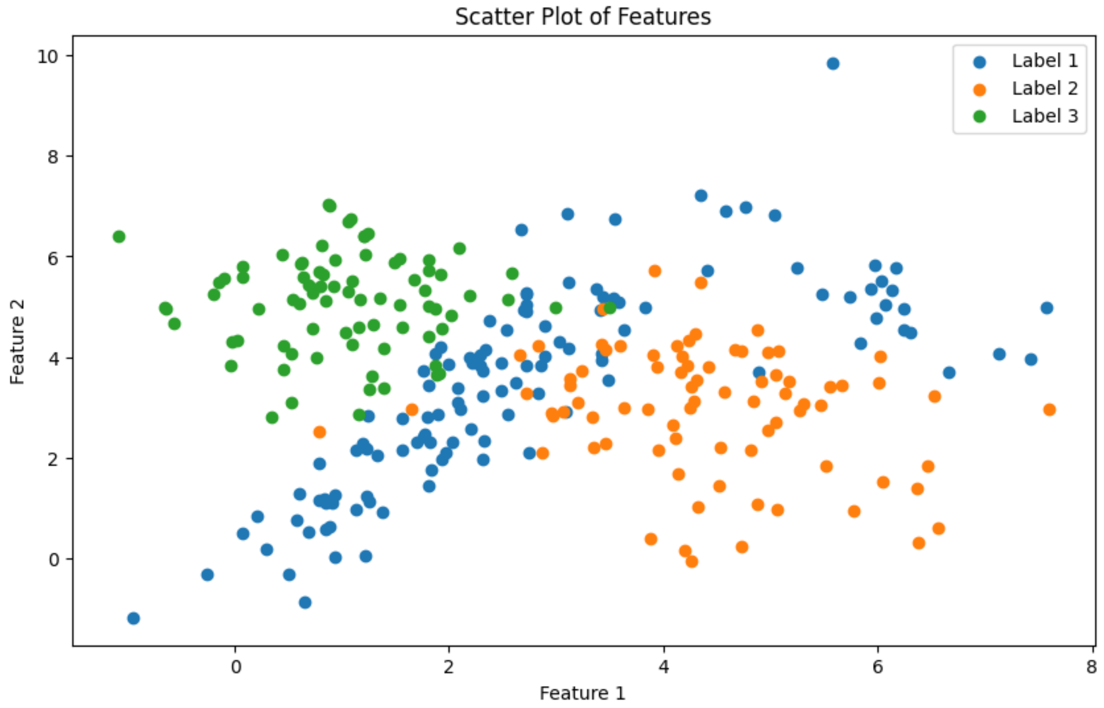
Pattern Recognition

Γιακουμόγλου Πασχάλης 10054

Γκύζης Γεώργιος 10107

Data Visualization and Split (Part A-C)



- Clear distribution of data points across two features.
- Overlapping regions suggest potential challenges for classification.
- Class 1: 42.86%, Class 2: 28.57%, Class 3: 28.57% (No heavy imbalances)
- Similar values for both features (no need for scaling, $var(feat1) \approx var(feat2)$)
- The data will be split 0.5-0.5
- The same datasets will be used for Parts A,B,C

Classifier Regions (Part A-C)

- Mesh Grid that covers the latent space of the features
- Classify each point of the grid
- Different coloring for each region and different shapes for training and test data

Part A: Bayes Gaussian Classifier

- The classifier assumes gaussian distribution of the features of each class
- Two cases are implemented: Same or different covariance matrices for each class
- Same covariance matrix: simpler model, danger of underfitting
- Different covariance matrix: complex model, danger of overfitting but can capture more complex patterns

Part A: Bayes Gaussian Classifier

- For the distribution estimation the maximum likelihood technique is followed
- For the Gaussian Distribution, the maximum likelihood estimators are:

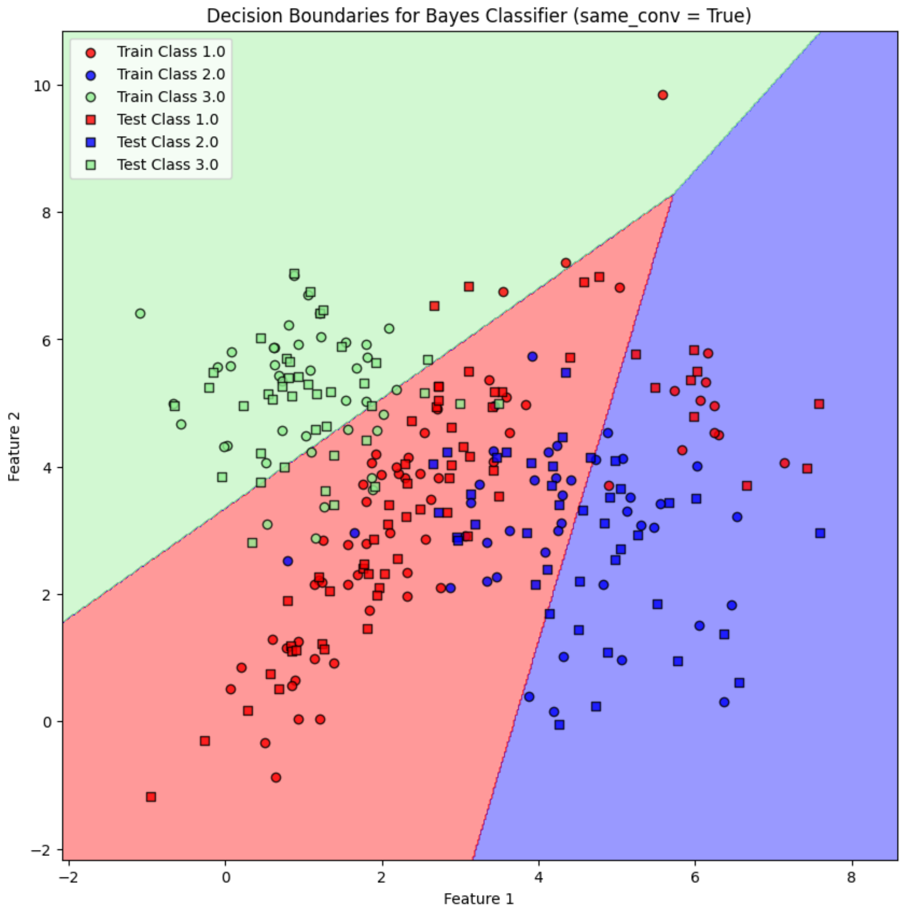
$$\hat{\mu}_n = \frac{1}{n}\sum_{j=1}^{n} x_j$$

$$\hat{V}_n = \frac{1}{n}\sum_{j=1}^{n}(x_j - \hat{\mu}_n)(x_j - \hat{\mu}_n)^T$$

Part A: Bayes Gaussian Classifier

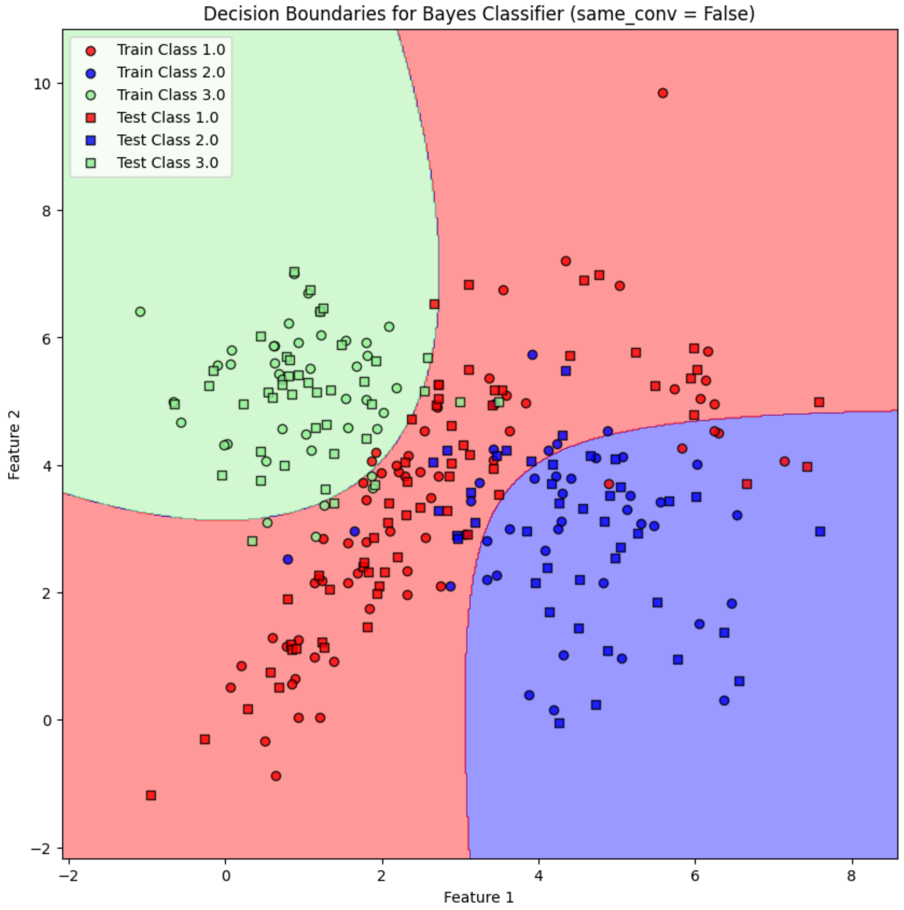
- In our implementation the classifier has similar methods to the sklearn (fit, predict, predict_proba)
- That ensures that our estimator could be used later with other methods of sklearn (cross_val, ensembles, etc.)

Part A: Bayes Gaussian Classifier



- Assuming the same covariance matrix, the classifier achieves 71% accuracy on the test set
- The same covariance matrix creates linear boundaries for each region
- The classifier is limited by the linear boundaries
- Cluster of points of the same class are incorrectly classified
- More complex classifier could potentially capture these patterns

Part A: Bayes Gaussian Classifier



- Assuming a different covariance matrix, the classifier achieves 85% accuracy on the test data
- The different covariance matrix creates non linear boundaries for each region
- The classifier can capture more intricate patterns in the data
- This is confirmed by the improved accuracy on the training set

Part B: K-Nearest Neighbors Classifier

- sk-learn implementation of the classifier
- Euclidean distance as a distance metric
- Lower of values of k-NN are prone to overfitting
- Higher values of k-NN usually can not capture complex patterns in the dataset

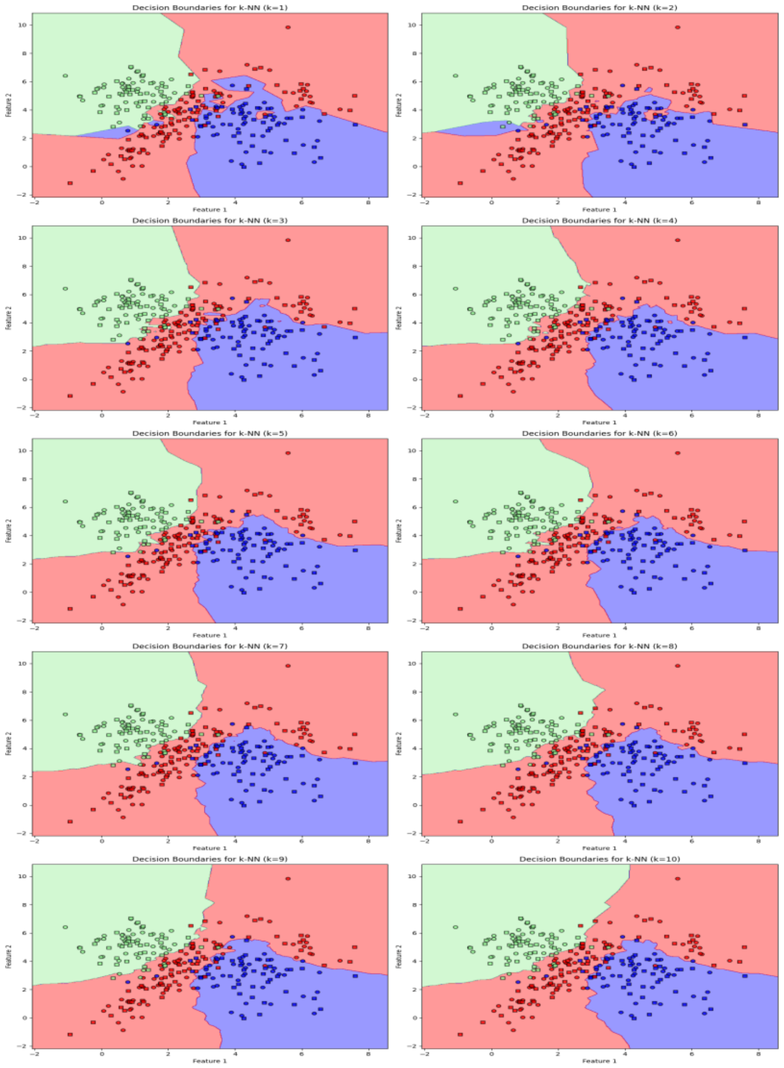
Part B: K-Nearest Neighbors Classifier

Results of the K-NN

Number of neighbor	Accuracy (Test Set)
1-NN	88.57%
2-NN	87.14%
3-NN	87.14%
4-NN	87.14%
5-NN	89.28%
6-NN	87.58%
7-NN	88.58%
8-NN	89.28%
9-NN	90.00%
10-NN	88.00%

- K=1: Complex decision boundary, likely overfitting.
- Slight decrease then plateau around 87.14% for K = 2 to K = 4.
- Increase to 89.23% at K = 5
- High but fluctuating trend, peaking at 90% for k=9

Part B: K-Nearest Neighbors Classifier



- K=1: Complex decision boundary, likely overfitting.
- K = 3 to K = 5: Smoother boundaries indicate improved generalization.
- K = 6 to K = 10: Continued smoothness with effective class separation, balanced bias and variance.
- Higher K values show no underfitting, maintaining model effectiveness.

Bayes vs KNN

- Comparing the accuracy scores, the k-NN classifier's best result (90% for k=9) outperforms the Gaussian Bayes classifier's accuracy of 85% with separate covariance matrices. This suggests that k-NN may be more adept at handling this particular dataset's intricacies without making strong distributional assumptions.

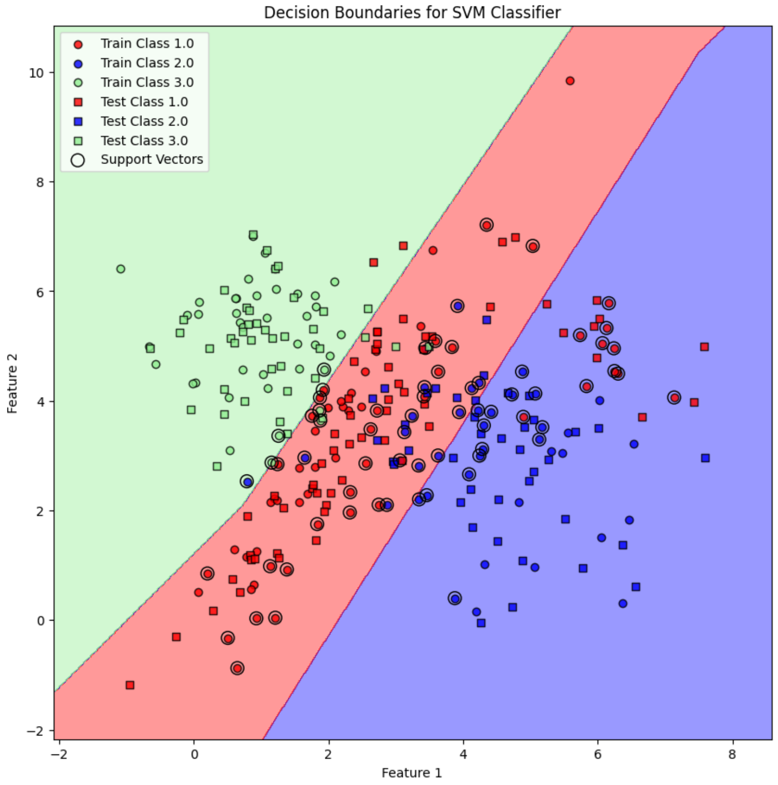
Part C: Linear SVMs

- Train a linear SVM classifier with GridSearch for optimal 'C’.
- 'C' parameter balances classifier simplicity and training performance.
- Higher 'C' values increase model complexity and cost of misclassification.
- Aim to find 'C' that fits well without causing overfitting.

Part C: Linear SVMs

- For the grid search we perform cross validation on the train set and use the model with the best average accuracy on the test set
- Best C: 100
- Accuracy on the test set: 0.8

Part C: Linear SVMs



- Notable class overlap and dense support vectors for classes 2 and 3.
- Linear SVM outperforms Bayes model with same covariance matrix.
- Class overlap hints at linear SVM's limitation in capturing data complexity.
- Suggests potential benefits of using non-linear kernels for complexity modeling.

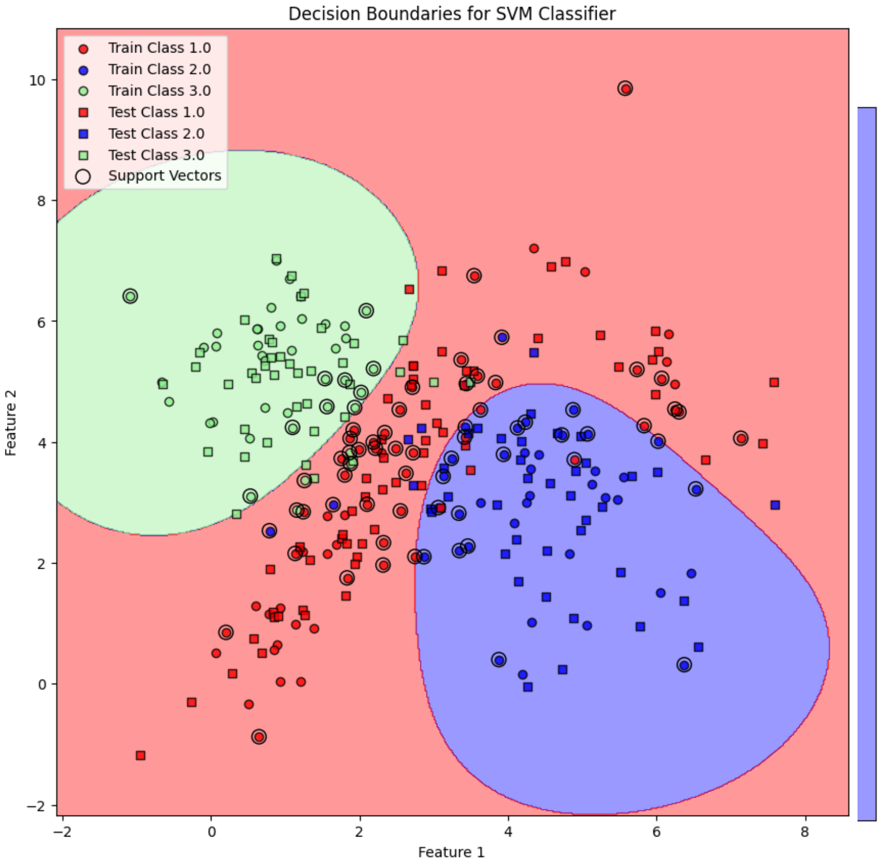
Part C: SVM with RBF kernel

- Perform GridSearch on SVM with RBF kernel for 'C' and 'gamma' hyperparameters.
- 'Gamma' determines the influence range of support vectors.
- High 'gamma': small influence radius, irregular boundaries, risk of overfitting.
- Low 'gamma': large influence radius, smoother boundaries, risk of underfitting.

Part C: SVM with RBF kernel

- For the grid search we perform cross validation on the train set and use the model with the best average accuracy on the test set
- Best score cv: 90%
- Best C: 1
- Best γ: 0.1
- Accuracy on the test set: 88.57%

Part C: SVM with RBF kernel



- Highly Complex decision Regions
- Well-tuned model with balanced decision boundaries.
- rbf-SVM vastly outperforms linear model.
- Fewer support vectors than linear SVM, capturing complex patterns.

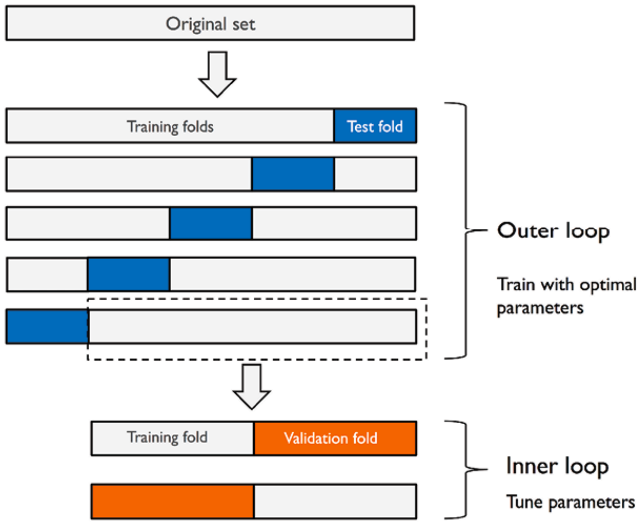
Overall

- Best k-NN classifier leads with 90% test accuracy.
- RBF kernel SVM follows closely at 88.57%.
- Gaussian Bayes with separate covariance matrices: 85% accuracy.
- Linear SVM shows lower performance at 80%.
- Gaussian Bayes with shared covariance matrix has the least at 72.42%.

Part D: Model Selection

- Cross-Validation Practice:
  - Perform cross-validation on training data.
  - Select model with best cross-validation accuracy.
  - Model that best fits in each validation fold, not model with least generalization error
- Nested Cross-Validation for Generalization:
  - Use nested cross-validation to estimate generalization error.
  - Outer loop divides data into training and testing folds.
  - Inner loop performs cross-validation on each training fold.
  - Average accuracy reported on outer testing sets.

Part D: Model Selection



https://vitalflux.com/python-nested-cross-validation-algorithm-selection/

- Purpose of Nested Cross-Validation:
  - Utilized for model selection, not hyperparameter tuning.
  - After model selection, conduct cross-validation on full training set for optimal hyperparameters.

Part D: Bayesian Optimization

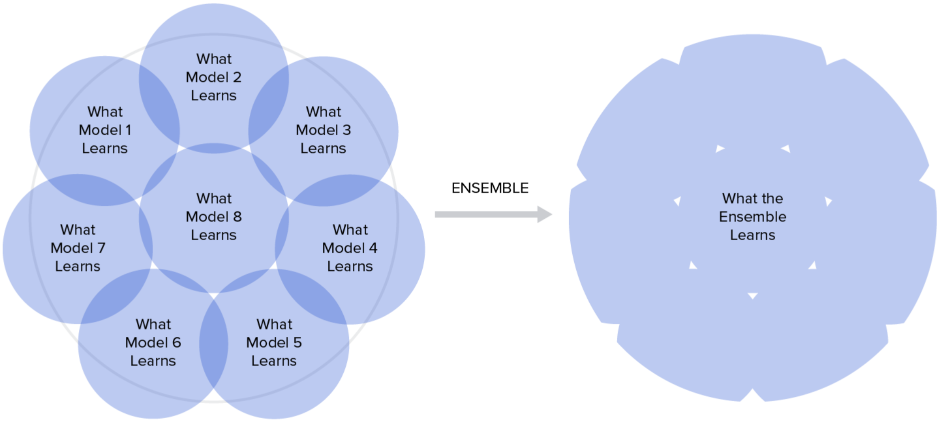
- Bayesian Optimization Overview:
  - Sequential strategy for globally optimizing expensive-to-evaluate black-box functions.
  - Treats objective functions as random function, using priors for initial behavior assumptions.
  - Updates priors to posteriors post-evaluation, guiding next query points.
- Application in Classifier Optimization:
  - Applied to classifiers as black-box functions.
  - Uses 5-fold cross-validation accuracy as the evaluation metric.
- Implementation Tool:
  - Utilized the scikit-optimize library for implementation.

Part D: Bayesian Vs Grid Search

- Efficiency: Bayesian optimization is more efficient as it requires fewer function evaluations. It intelligently chooses the next point to evaluate based on past evaluations.
- Handling Complex Spaces: Better suited for optimizing over high-dimensional and complex spaces where grid search becomes impractical.
- Flexible and Informative: Adjusts exploration and exploitation dynamically, making it more informative for decision-making compared to the exhaustive nature of grid search.

Part D: Ensembles

- Ensembles:
  - Combine multiple models to enhance prediction accuracy.
  - Leverage strengths and mitigate weaknesses of individual models.
  - Most Decorated Model Family in Kaggle competitions



https://www.toptal.com/machine-learning/ensemble-methods-kaggle-machine-learn

Part D: Boosting

- Boosting Overview:
  - Sequentially trains models, focusing more on instances misclassified by previous models.
  - Weighs each model and instance to prioritize correcting errors, enhancing overall performance.
- AdaBoost
- XGBoost
  - Solves hyperparameter tuning complexity in boosting by optimizing a loss function across all models.
  - Known for efficiency and performance, incorporating techniques like parallel computing and regularization. Won recognition for its effectiveness in a Kaggle competition.

Part D: Bagging

- Bagging Overview:
  - Utilizes bootstrapping to create overlapping data subsets.
  - Aggregates models' predictions using mean, median, mode, or weighted metrics.
- RandomForest:
  - Applies bagging with decision trees, using random subsets of attributes at each node.
  - Ensures diversity among trees, enhancing performance.
  - Effective for feature selection, offers good performance with low variance and training time.

Part D: Voting Classifiers

- Voting Classifier Concept:
  - Combines predictions from multiple models.
  - Utilizes either 'hard' voting (majority rule) or 'soft' voting (weighted probabilities)
- Advantages:
  - Leverages strengths of various models.
  - Can improve prediction robustness and accuracy.

Part D: Stacking Classifiers

- Stacking Overview:
  - Combines multiple models' predictions as input for a final meta-model.
  - Enhances predictive accuracy beyond individual models.
- Training Methodology:
  - Base models are trained on a part of the dataset.
  - Their out-of-sample predictions form a new dataset to train the meta-model.
  - This methodology ensures that both the training and testing dataset come from the same distribution or the the meta classifier doesn’t prefer the most overfitted model

Part D: Stacking Classifiers

- Limitations of the sk-learn implementation:
  - Can not use engineered features as input to the meta classifier
  - Computational inefficiency: If we want to evaluate base models we must retrain them
  - Difficult to fine tune: The more base models are used the more difficult it is to fine tune them

Part D: Final Model

- To select a model we used an outer loop of 2 folds and an inner loop of 5 folds
- The following models were tried: SVM (Bayes search), MLP (Grid search), 1d-CNN (Grid search), XGBoost (Bayes search), RandomForest (Bayes search), HistGradBoosting (bayes search), Stacking (Grid Search), AdaBoost (Bayes search)
- Our winner was a stacking classifier with 10 base models and a soft voting classifier with (xgb, MLP, adaboost, SVM) as a meta classifier. It achieved an accuracy of 86% in the outer folds.

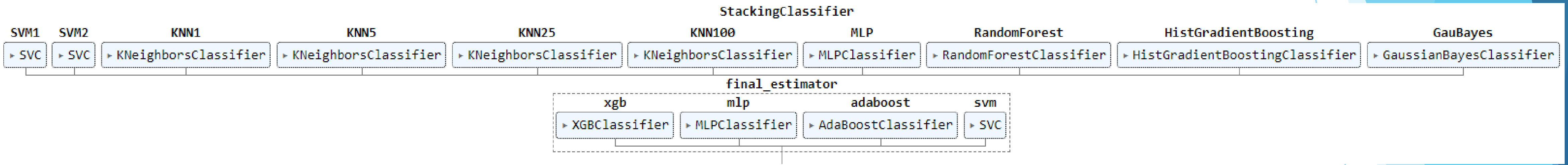
Results

Model	Outer Fold Accuracy
SVM	84.14%
MLP	80%
1d-CNN	83.28%
XGBoost	75.44%
RandomForest	79.13%
HistGradBoosting	77.58%
Adaboost	71%
Stacking (meta = xgb)	84.12%
Stacking (meta = MLP)	86.01%
Stacking (meta = AdaBoost)	85.13%
Stacking (meta = SVM)	86.11%
Stacking (meta = {xgb, MLP, AdaBoost, SVM})	86.24%

Part D: Observations

- General:
  - PCA, t-SNE and scaling did not improve results
- Stacking
  - Trying different parameters sets for the base models did not lead to improvement (due to limited computational resources not many sets were tried)
  - Fine tuning the meta-model using grid search provided improved performance in the outer loop
  - Fine tuning the models of the meta voting classifier did not lead to improved performance

Part D: Final Model



- Model Tuning Observations:

- Fine-tuning base models increased time complexity significantly without noticeable improvement.
- Experimented with different meta-classifiers; soft voting classifier yielded the best results.
- No significant gains from fine-tuning the models within the voting classifier.

THE END