IHI

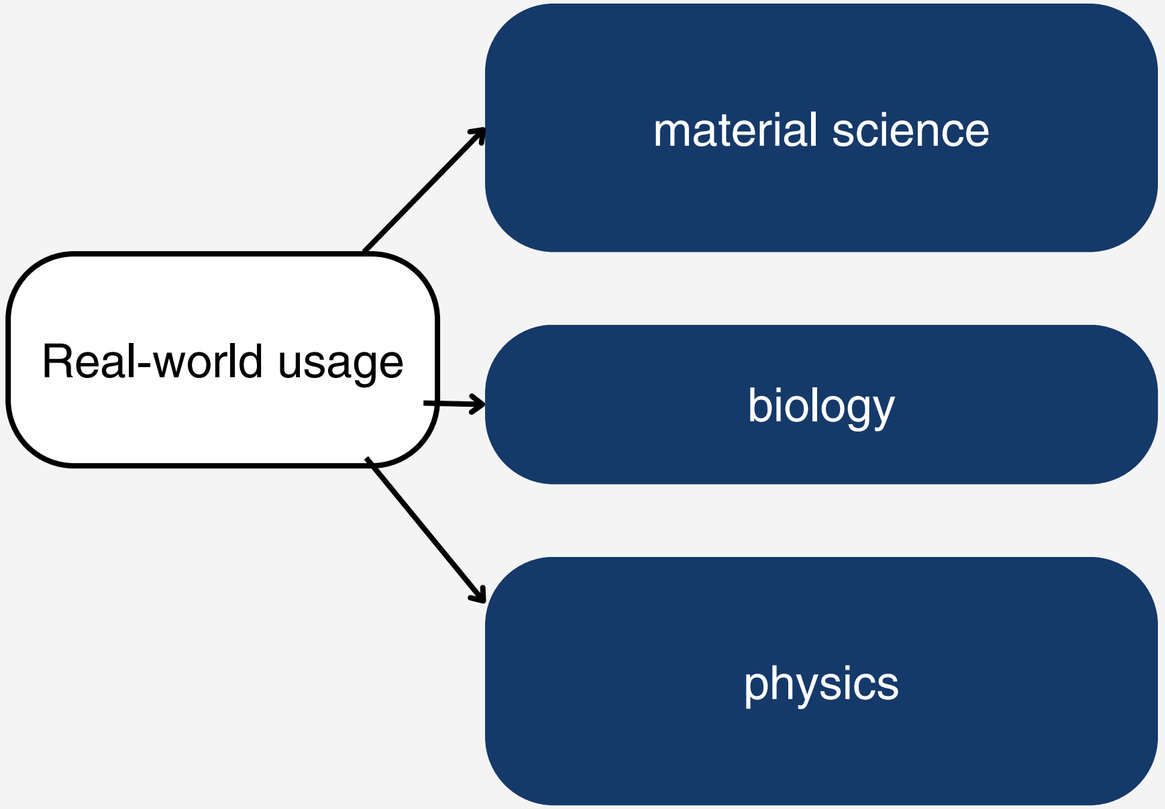
Yoroshiku onegaishimasu!!!

Classification of Diffusion Types Using Machine Learning

By Gargi Rathi

Objective

Identifying diffusion types based on features like distance and time using machine learning.



About the Dataset

- Dataset: DiffData_classification_noise.csv
- Columns: Time, Distance (float), Diffusion_type (categorical)
- Purpose: Diffusion_type represents different diffusion behaviors
- Total Rows: 981

time	distance	Diffusion_type
477.3006134969325	0.012937787913532676	Db
191.41104294478527	0.030020147087505136	Dgb
527.6073619631902	0.029474894249549414	Db
251.53374233128835	0.00535860087311998	Db
404.9079754601227	0.020115335680653493	Db
251.53374233128835	0.04448751832151919	Dgb
96.93251533742331	0.02298607533538002	Db
375.4601226993865	0.029903882530479133	Db

Data Preprocessing steps

Converted categorical labels into binary classes

Dgb = 1, others = 0

Feature selection

Used only distance and time as input

Standardized data

Used StandardScaler for better performance

Models Used

IHI

K-Nearest Neighbors (KNN)

Simple, intuitive, but struggles with high-dimensional data

Support Vector Machine (SVM)

Effective with small datasets, required fine-tuning of kernel and gamma

Random Forest

Handled noise well, provided feature importance, but required hyperparameter tuning

KNN

K-Nearest Neighbors

Load Dataset

- Read CSV
- dataset.describe().T (Check statistics)

Preprocessing

- Create binary target column (1 for "Dgb", 0 otherwise)
- Select features: distance, time (xInput) → Target: yTarget
- Train-test split (80-20, random_state=0)
- Standardization using StandardScaler()

Model Training & Evaluation

- KNN Classifier
  - n_neighbors=5
  - Train (clf.fit())
  - Test & Evaluate (confusion_matrix, accuracy_score, classification_report)
  - Cross-validation (cross_val_score(), 5-fold)
- SVM Classifier
  - Train with default SVC()
  - Test & Evaluate

Hyperparameter Tuning (GridSearchCV)

- Parameters tuned:
  - n_neighbors (3, 5, 7, 9)
  - Weighting (uniform, distance)
  - Distance metrics (euclidean, manhattan, minkowski)
- Best model selected → Retrained & evaluated

Random

Random forest approach

IHI

Load Dataset

- Read CSV
- dataset.describe().T (Check statistics)

Preprocessing

- Create binary target column (1 for "Dgb", 0 otherwise)
- Select features: distance, time (xInput) → Target: yTarget
- Train-test split (80-20, random_state=0)
- Standardization using StandardScaler()

Model Training & Evaluation

- Random Forest Classifier
  - Train model (RandomForestClassifier())
  - Test & Evaluate (confusion_matrix, accuracy_score, classification_report)
  - Cross-validation (cross_val_score(), 5-fold)

Hyperparameter Tuning (RandomizedSearchCV)

- Parameters tuned:
  - n_estimators (No. of trees)
  - max_depth (Tree depth)
- Best model selected → Retrained & evaluated

IHI

SVM

Support Vector Machine

Load Dataset

- Read CSV
- dataset.describe().T (Check statistics)

Preprocessing

- Create binary target column (1 for "Dgb", 0 otherwise)
- Select features: distance, time (xInput) → Target: yTarget
- Train-test split (80-20, random_state=0)
- Standardization using StandardScaler()

Baseline Model (SVM Classifier)

- Train model (SVC(), default settings)
- Test & Evaluate (confusion_matrix, accuracy_score, classification_report)
- Cross-validation (cross_val_score(), 5-fold)

Hyperparameter Tuning (GridSearchCV & RandomizedSearchCV)

- GridSearchCV
  - Parameters tuned:
    - C (Regularization: 0.1, 1, 10)
    - kernel (linear, rbf, poly)
    - gamma (scale, auto)
  - Best model selected → Retrained & evaluated
- RandomizedSearchCV
  - Random sampling of:
    - C (0.1 to 10, uniform)
    - gamma (log scale)
  - Iterations = 18 (computational efficiency)
  - Best model selected → Retrained & evaluated

Model Comparison

IHI

K-Nearest Neighbors (KNN)

Best Hyperparameters:
{'metric': 'euclidean', 'n_neighbors': 5, 'weights': 'distance'}

Best SVM Accuracy: 0.93

```
Confusion Matrix:
 [[86 12]
 [ 4 94]]

Accuracy: 0.9183673469387755

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.88      0.91        98
           1       0.89      0.96      0.92        98

    accuracy                           0.92       196
   macro avg       0.92      0.92      0.92       196
weighted avg       0.92      0.92      0.92       196


Cross-Validation Scores: [0.92356688 0.92356688 0.94267516 0.9044586  0.91025641]
Final CV Score: 0.92
```

Support Vector Machine (SVM)

Best Hyperparameters:
{'C': 10, 'gamma': 1, 'kernel': 'rbf'}

Best SVM Accuracy: 0.93

```
Confusion Matrix:
 [[81 17]
 [ 0 98]]


accuracy:  0.9132653061224489


              precision    recall  f1-score   support

           0       1.00      0.83      0.91        98
           1       0.85      1.00      0.92        98

    accuracy                           0.91       196
   macro avg       0.93      0.91      0.91       196
weighted avg       0.93      0.91      0.91       196

[0.92356688 0.91719745 0.93630573 0.89808917 0.91025641]
Final CV Score:  0.92
```

Random Forest

Best Hyperparameters:
{'max_depth': 6, 'n_estimators': 479}

Best SVM Accuracy: 0.90

```
Confusion Matrix:
 [[86 12]
 [ 7 91]]


accuracy:  0.9030612244897959


              precision    recall  f1-score   support

           0       0.92      0.88      0.90        98
           1       0.88      0.93      0.91        98

    accuracy                           0.90       196
   macro avg       0.90      0.90      0.90       196
weighted avg       0.90      0.90      0.90       196

[0.88535032 0.89808917 0.92993631 0.9044586  0.8974359 ]
Final CV Score:  0.90
```

Final Conclusion

Best Model Selection

SVM (Based on highest accuracy & evaluation metrics)

Key Observations:

- SVM outperforms KNN & Random Forest, achieving the highest accuracy of 0.93.
- Best Hyperparameters for SVM: {'C': 10, 'gamma': 1, 'kernel': 'rbf'}.
- RandomizedSearchCV improved Random Forest tuning efficiency, reducing computational cost.
- KNN performed well but was highly sensitive to the choice of distance metric (euclidean).

SVM (with RBF kernel) proved to be the best choice for classifying diffusion types based on distance & time.

link-(https://www.canva.com/design/DAGecaj38kY/gZYhw4u0L47cizb2tLf0EQ/edit?utm_content=DAGecaj38kY&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton)