30/09/2025

SVM Classification

Method	Best Parameters	Best CV Accuracy	Precision (weighted)	Recall (weighted)	F1-score (weighted)	Confusion Matrix
Svm **4 features[1]**	{'svm__C': 10, 'svm__class_weight': None, 'svm__gamma': 0.001}	**0.743**	**0.71**	**0.70**	**0.70**	**[[3 2]** **[1 4]]**
Svm **15 features[2]**	{'svm__C': 100, 'svm__class_weight': None, 'svm__gamma': 0.01}	**0.921**	**0.922**	**0.90**	**0.90**	**[[5 0]** **[1 4]]**
Svm + RFE + folding **15 features[3]**		**0.955 (mean)**	**0.964955 (mean)**	**0.955955 (mean)**	**0.9555955 (mean)**	**[[24 0]** **[0 24]]**

Change that i made :

Started with the normal SVM[1] with 5 features , got accuracy of **74%**, and made change to the svm model that added an **Recursive Feature Elimination** to find the most important features and we got **15 new features**, also used **folding of data set** test all the parts of the data, and we got **95%(mean)** for that model[3]. Then we used that higher priority fetures in the normal svm model [2], and it able to perform in an accuracy of **92%.**

Methode 1 :

```
selected_features = [
    "'Cerebellar Vermal Lobules VI-VII'",
```

"'Left PCu precuneus'",
"'Right LiG lingual gyrus'",
"'Right PoG postcentral gyrus'",
"'Right MPoG postcentral gyrus medial segment'"
]

Method 2:

selected_features = [
"'4th Ventricle'",
"'Left Lateral Ventricle'",
"'Cerebellar Vermal Lobules VI-VII'",
"'Right AnG angular gyrus'",
"'Right Calc calcarine cortex'",
"'Left Calc calcarine cortex'",
"'Right Cun cuneus'",
"'Left Cun cuneus'",
"'Right FO frontal operculum'",
"'Right LiG lingual gyrus'",
"'Left MCgG middle cingulate gyrus'",
"'Right MFG middle frontal gyrus'",
"'Left MOrG medial orbital gyrus'",
"'Right PoG postcentral gyrus'",
"'Right SMG supramarginal gyrus'"
]

Methode 3:

Result:

Evaluation Metrics per Fold:

Fold 1:

Accuracy: 1.0

Precision: 1.0

Recall: 1.0

F1-score: 1.0

Confusion Matrix:

[[5 0]

[0 5]]

Fold 2:

Accuracy: 1.0

Precision: 1.0

Recall: 1.0

F1-score: 1.0

Confusion Matrix:

[[5 0]

[0 5]]

Fold 3:

Accuracy: 1.0

Precision: 1.0

Recall: 1.0

F1-score: 1.0

Confusion Matrix:

[[5 0]

[0 5]]

Fold 4:

Accuracy: 0.8888888888888888

Precision: 0.911111111111111

Recall: 0.8888888888888888

F1-score: 0.8888888888888888

Confusion Matrix:

[[4 1]

[0 4]]

Fold 5:

Accuracy: 0.8888888888888888

Precision: 0.911111111111111

Recall: 0.8888888888888888

F1-score: 0.8888888888888888

Confusion Matrix:

[[4 0]

[1 4]]

--- Overall Metrics ---

Mean Accuracy: 0.9555555555555555

Mean Precision: 0.9644444444444444

Mean Recall: 0.9555555555555555

Mean F1-score: 0.9555555555555555

Code of Method 3 [3]:

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.feature_selection import RFE
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix
import joblib
import warnings
import numpy as np

warnings.filterwarnings("ignore")
```

```
# ---------- Load Dataset ----------
file_name = "VBM data.xlsx"
df = pd.read_excel(file_name)
df.columns = df.columns.str.strip().str.replace("'", "")

# ---------- Features & Target ----------
target_col = "Group"
non_features = ['Group', 'Age', 'Gender']
feature_columns = [col for col in df.columns if col not in non_features]

X = df[feature_columns]
y = df[target_col]

# ---------- Scale Features ----------
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# ---------- RFE with SVM ----------
svm_estimator = SVC(kernel='linear', C=1, random_state=42)
rfe = RFE(estimator=svm_estimator, n_features_to_select=15, step=1)
rfe.fit(X_scaled, y)

# Top 15 Features
top_features = [f for f, s in zip(feature_columns, rfe.support_) if s]
print("Top 15 Features selected by RFE with SVM:")
for f in top_features:
    print("-", f)

# Train SVM on selected features
X_rfe_scaled = scaler.fit_transform(X[top_features])
svm_best = SVC(kernel='linear', C=1, random_state=42)
```

```
svm_best.fit(X_rfe_scaled, y)

# ---------- Cross-validation with detailed metrics ----------
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

accuracies, precisions, recalls, f1s = [], [], [], []

print("\nEvaluation Metrics per Fold:")

for fold, (train_idx, test_idx) in enumerate(cv.split(X_rfe_scaled, y), 1):
    X_train, X_test = X_rfe_scaled[train_idx], X_rfe_scaled[test_idx]
    y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

    svm_fold = SVC(kernel='linear', C=1, random_state=42)
    svm_fold.fit(X_train, y_train)
    y_pred = svm_fold.predict(X_test)

    acc = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred, average='weighted', zero_division=0)
    rec = recall_score(y_test, y_pred, average='weighted', zero_division=0)
    f1 = f1_score(y_test, y_pred, average='weighted', zero_division=0)
    cm = confusion_matrix(y_test, y_pred)

    accuracies.append(acc)
    precisions.append(prec)
    recalls.append(rec)
    f1s.append(f1)

    print(f"\nFold {fold}:")
    print("Accuracy:", acc)
    print("Precision:", prec)
    print("Recall:", rec)
```

```
    print("F1-score:", f1)
    print("Confusion Matrix:\n", cm)

# ---------- Overall Metrics ----------
print("\n--- Overall Metrics ---")
print("Mean Accuracy:", np.mean(accuracies))
print("Mean Precision:", np.mean(precisions))
print("Mean Recall:", np.mean(recalls))
print("Mean F1-score:", np.mean(f1s))

# ---------- Save Model, Scaler, Features ----------
joblib.dump(svm_best, "svm_rfe_model.joblib")
joblib.dump(scaler, "scaler_rfe.joblib")
joblib.dump(top_features, "rfe_features.joblib")
print("\nSVM model, scaler, and feature list saved successfully!")
```