Stacked_ensemble

August 29, 2025

[2]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV,␣
 ↪StratifiedKFold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score, roc_auc_score,␣
 ↪roc_curve,auc,confusion_matrix
)
import matplotlib.pyplot as plt
from sklearn.ensemble import StackingClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
```

[2]:

```
# 1. Load dataset
# wdbc.data does not have headers, so we define them
columns = ["ID", "Diagnosis"] + [f"feature_{i}" for i in range(1, 31)]
data = pd.read_csv("wdbc.data", header=None, names=columns)
```

[3]:

```
# 2. Prepare features and target
X = data.drop(["ID", "Diagnosis"], axis=1)
y = data["Diagnosis"].map({"M": 1, "B": 0})  # Malignant=1, Benign=0
```

[4]:

```
# 4. Preprocessor (scaling not needed for trees, but kept for pipeline␣
 ↪consistency)
num_features = X.columns.tolist()
preprocessor = ColumnTransformer(
    transformers=[("scale", StandardScaler(), num_features)],
    remainder="drop"
)
```

[5]:

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

[6]:

```
# --- 1. Define base models ---
base_estimators = [
    ("svm", SVC(probability=True, random_state=42)),
    ("nb", GaussianNB()),
    ("dt", DecisionTreeClassifier(random_state=42))
]
```

[7]:

```
# --- 2. Define Stacking Classifier with Logistic Regression as final estimator ---
stack = StackingClassifier(
    estimators=base_estimators,
    final_estimator=LogisticRegression(max_iter=1000, random_state=42),
    passthrough=False
)
```

[8]:

```
# --- 3. Hyperparameter Grid for final estimator ---
param_grid = {
    "final_estimator": [
        LogisticRegression(max_iter=1000, random_state=42),
        RandomForestClassifier(n_estimators=100, random_state=42)
    ]
}
```

[9]:

```
# --- 4. Grid Search ---
stack_grid = GridSearchCV(
    stack,
    param_grid,
    cv=3,
    scoring={"accuracy": "accuracy", "f1": "f1_macro"},
    refit="accuracy",
    n_jobs=-1
)
stack_grid.fit(X_train, y_train)
```

[9]:

```
GridSearchCV(cv=3,
             estimator=StackingClassifier(estimators=[('svm',
                                                        SVC(probability=True,
                                                            random_state=42)),
                                                       ('nb', GaussianNB()),
                                                       ('dt',
DecisionTreeClassifier(random_state=42))],
final_estimator=LogisticRegression(max_iter=1000,
random_state=42)),
             n_jobs=-1,
             param_grid={'final_estimator': [LogisticRegression(max_iter=1000,
random_state=42),
RandomForestClassifier(random_state=42)]},
             refit='accuracy',
             scoring={'accuracy': 'accuracy', 'f1': 'f1_macro'})
```

[10]:

```
results = []
for params, acc, f1 in zip(
    stack_grid.cv_results_["params"],
    stack_grid.cv_results_["mean_test_accuracy"],
    stack_grid.cv_results_["mean_test_f1"]
):
    final_estimator = type(params["final_estimator"]).__name__
    results.append([
        "SVM, Naïve Bayes, Decision Tree",
        final_estimator,
        f"{acc:.4f}",
        f"{f1:.4f}"
    ])
```

[11]:

```
# --- 6. Convert to DataFrame ---
stack_table = pd.DataFrame(
    results,
    columns=["Base Models", "Final Estimator", "Accuracy", "F1 Score"]
)

print("Stacked Ensemble Model")
print("Hyperparameter Trials")
print("Table 6: Stacked Ensemble - Hyperparameter Tuning")
print(stack_table)
```

```
Stacked Ensemble Model
Hyperparameter Trials
Table 6: Stacked Ensemble - Hyperparameter Tuning
                        Base Models         Final Estimator Accuracy F1 Score
0  SVM, Naïve Bayes, Decision Tree      LogisticRegression   0.9494   0.9452
1  SVM, Naïve Bayes, Decision Tree  RandomForestClassifier   0.9363   0.9315
```

[12]:

```
# Train the Stacked Ensemble: SVM + Decision Tree + KNN
stack_clf_knn = StackingClassifier(
    estimators=[("svm", SVC(probability=True, random_state=42)),
                ("dt", DecisionTreeClassifier(random_state=42)),
                ("knn", KNeighborsClassifier())],
    final_estimator=LogisticRegression(max_iter=1000, random_state=42),
    cv=5,
    n_jobs=-1
)
```

```python

stack_clf_knn.fit(X_train, y_train)
y_val_pred = stack_clf_knn.predict(X_test)

acc = accuracy_score(y_test, y_val_pred)
f1 = f1_score(y_test, y_val_pred, average="weighted")

# Single row result
stack_table_knn = pd.DataFrame([{
    "Base Models": "SVM, Decision Tree, KNN",
    "Final Estimator": "Logistic Regression",
    "Accuracy / F1 Score": f"{acc:.3f} / {f1:.3f}"
}])

print("Stacked Ensemble Model")
print("Hyperparameter Trials")
print("Table 6: Stacked Ensemble - Hyperparameter Tuning")
print(stack_table_knn)
```

```
Stacked Ensemble Model
Hyperparameter Trials
Table 6: Stacked Ensemble - Hyperparameter Tuning
               Base Models      Final Estimator Accuracy / F1 Score
0  SVM, Decision Tree, KNN  Logistic Regression       0.939 / 0.938
```