Liang_Assign5

June 30, 2024

[78]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings("ignore")
```

[31]:
```
df = pd.read_csv("heart_dataset-1.csv")
```

[32]:
```
df.shape
```

[32]:
```
(918, 12)
```

[33]:
```
df.head()
```

[33]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	\
0	40	M	ATA	140	289	0	Normal	172	
1	49	F	NAP	160	180	0	Normal	156	
2	37	M	ATA	130	283	0	ST	98	
3	48	F	ASY	138	214	0	Normal	108	
4	54	M	NAP	150	195	0	Normal	122	

	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	N	0.0	Up	0
1	N	1.0	Flat	1
2	N	0.0	Up	0
3	Y	1.5	Flat	1
4	N	0.0	Up	0

[34]:
```
df.columns.values
```

[34]:
```
array(['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol',
       'FastingBS', 'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak',
       'ST_Slope', 'HeartDisease'], dtype=object)
```

[35]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Age             918 non-null    int64  
 1   Sex             918 non-null    object 
 2   ChestPainType   918 non-null    object 
 3   RestingBP       918 non-null    int64  
 4   Cholesterol     918 non-null    int64  
 5   FastingBS       918 non-null    int64  
 6   RestingECG      918 non-null    object 
 7   MaxHR           918 non-null    int64  
 8   ExerciseAngina  918 non-null    object 
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object 
 11  HeartDisease    918 non-null    int64  
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
```

[36]:
```
for i in [1,2,6,8,10]:
    print(df.iloc[:,i].unique())
```

```
['M' 'F']
['ATA' 'NAP' 'ASY' 'TA']
['Normal' 'ST' 'LVH']
['N' 'Y']
['Up' 'Flat' 'Down']
```

[37]:
```
df_encoded = pd.get_dummies(df, columns=["Sex", "ChestPainType", "RestingECG",␣
↪"ExerciseAngina", "ST_Slope"])

df_encoded
```

[37]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease	\
0	40	140	289	0	172	0.0	0	
1	49	160	180	0	156	1.0	1	
2	37	130	283	0	98	0.0	0	
3	48	138	214	0	108	1.5	1	
4	54	150	195	0	122	0.0	0	
..	…	…	…	…	…	…	…	
913	45	110	264	0	132	1.2	1	
914	68	144	193	1	141	3.4	1	
915	57	130	131	0	115	1.2	1	
916	57	130	236	0	174	0.0	1	
917	38	138	175	0	173	0.0	0	

	Sex_F	Sex_M	ChestPainType_ASY	…	ChestPainType_NAP	\

```
0    False   True              False  …              False
1     True  False              False  …               True
2    False   True              False  …              False
3     True  False               True  …              False
4    False   True              False  …               True
..     …     …                 …  …                 …
913  False   True              False  …              False
914  False   True               True  …              False
915  False   True               True  …              False
916   True  False              False  …              False
917  False   True              False  …               True

     ChestPainType_TA  RestingECG_LVH  RestingECG_Normal  RestingECG_ST  \
0               False           False               True          False
1               False           False               True          False
2               False           False              False           True
3               False           False               True          False
4               False           False               True          False
..                …               …                  …             …
913              True           False               True          False
914             False           False               True          False
915             False           False               True          False
916             False            True              False          False
917             False           False               True          False

     ExerciseAngina_N  ExerciseAngina_Y  ST_Slope_Down  ST_Slope_Flat  \
0                True             False          False          False
1                True             False          False           True
2                True             False          False          False
3               False              True          False           True
4                True             False          False          False
..                …                …              …             …
913              True             False          False           True
914              True             False          False           True
915             False              True          False           True
916              True             False          False           True
917              True             False          False          False

     ST_Slope_Up
0           True
1          False
2           True
3          False
4           True
..            …
913        False
914        False
```

```
915        False
916        False
917         True

[918 rows x 21 columns]
```

[59]:
```
from sklearn.model_selection import cross_val_score
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import LinearSVC
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

0.1 Problem 1

[56]:
```
dfX = df_encoded.loc[:, df_encoded.columns != 'HeartDisease']
dfy = df_encoded.loc[:, df_encoded.columns == 'HeartDisease'].values.ravel()
X = dfX.values
y = dfy

def eval_classifier(_clf, _X, _y):
    accuracies = []
    kf = StratifiedKFold(n_splits=10, shuffle=True, random_state=0)
    for train_index, test_index in kf.split(_X, _y):
        _clf.fit(_X[train_index], _y[train_index])
        y_pred = _clf.predict(_X[test_index])
        accuracies += [accuracy_score(_y[test_index], y_pred)]
    return np.array(accuracies)

acc = eval_classifier(GaussianNB(),
                      X, y)

print(f'Naive Bayes CV accuracy={np.mean(acc):.2f} {chr(177)}{np.std(acc):.3f}')
```

```
Naive Bayes CV accuracy=0.86 ±0.034
```

[60]:
```
acc = eval_classifier(LinearSVC(dual=False, tol=10),
                      X, y)
print(f'Linear SVC CV accuracy={np.mean(acc):.2f} {chr(177)}{np.std(acc):.3f}')
```

```
Linear SVC CV accuracy=0.45 ±0.002
```

[61]:
```
acc = eval_classifier(MLPClassifier(hidden_layer_sizes=(50,100),max_iter=500),
                      X, y)
print(f'Neural Network CV accuracy={np.mean(acc):.2f} {chr(177)}{np.std(acc):.
↪3f}')
```

```
Neural Network CV accuracy=0.82 ±0.052
```

[62]:
```
acc = eval_classifier(DecisionTreeClassifier(),
                      X, y)
print(f'Decision Tree CV accuracy={np.mean(acc):.2f} {chr(177)}{np.std(acc):.
 ↪3f}')
```

```
Decision Tree CV accuracy=0.79 ±0.046
```

[63]:
```
acc = eval_classifier(RandomForestClassifier(n_estimators=200, max_depth=5,␣
 ↪random_state=0, n_jobs=4),
                      X, y)
print(f'Random Forest CV accuracy={np.mean(acc):.2f} {chr(177)}{np.std(acc):.
 ↪3f}')
```

```
Random Forest CV accuracy=0.88 ±0.039
```

0.2 Problem 2

[40]:
```
mlp_weak = MLPClassifier(hidden_layer_sizes=(3, 3), max_iter=30, tol=1e-1)
dtc_weak = DecisionTreeClassifier(max_depth=5, max_features=5)
```

[127]:
```
gnb_ensemble = [GaussianNB() for _ in range(100)]
svc_ensemble = [SVC(kernel='linear', probability=True, max_iter=1000) for _ in␣
 ↪range(100)]
mlp_ensemble = [MLPClassifier(hidden_layer_sizes=(3, 3), max_iter=30, tol=1e-1)␣
 ↪for _ in range(100)]
dtc_ensemble = [DecisionTreeClassifier(max_depth=5, max_features=5) for _ in␣
 ↪range(100)]

def eval_first_classifier(ensemble, X, y):
    first_classifier = ensemble[0]
    first_classifier.fit(X, y)
    score = first_classifier.score(X, y)
    return score

gnb_first_score = eval_first_classifier(gnb_ensemble, X, y)
svc_first_score = eval_first_classifier(svc_ensemble, X, y)
mlp_first_score = eval_first_classifier(mlp_ensemble, X, y)
dtc_first_score = eval_first_classifier(dtc_ensemble, X, y)


ensemble_results = {
    "Classifier": ["GaussianNB", "LinearSVC", "MLPClassifier",␣
 ↪"DecisionTreeClassifier"],
    "First Classifier Score": [gnb_first_score, svc_first_score,␣
 ↪mlp_first_score, dtc_first_score]
}
```

[128]:
```
ensemble_results
```

[128]:
```
{'Classifier': ['GaussianNB',
  'LinearSVC',
  'MLPClassifier',
  'DecisionTreeClassifier'],
 'First Classifier Score': [0.8616557734204793,
  0.5729847494553377,
  0.4466230936819172,
  0.8812636165577342]}
```

0.3 Problem 3

[112]:
```
import random
from sklearn.utils import resample

def ensemble_fit(ensemble, X, y):
    for clf in ensemble:
        # Generate a bootstrap sample
        X_sample, y_sample = resample(X, y, replace=True, n_samples=int(0.8 *␣
 ↪len(X)))
        # Fit the classifier on the bootstrap sample
        clf.fit(X_sample, y_sample)


def report_first_classifier_performance(ensemble, X, y):
    first_clf = ensemble[0]
    y_pred = first_clf.predict(X)
    accuracy = accuracy_score(y, y_pred)
    print(f"Accuracy of the first classifier in the ensemble: {accuracy:.2f}")


ensemble_fit(gnb_ensemble, X, y)


print("GaussianNB Ensemble:")
report_first_classifier_performance(gnb_ensemble, X, y)
```

```
GaussianNB Ensemble:
Accuracy of the first classifier in the ensemble: 0.84
```

[113]:
```
ensemble_fit(svc_ensemble, X, y)


print("\nSVC Ensemble:")
report_first_classifier_performance(svc_ensemble, X, y)
```

```
SVC Ensemble:
Accuracy of the first classifier in the ensemble: 0.61
```

[79]:
```
ensemble_fit(mlp_ensemble, X, y)

print("\nMLPClassifier Ensemble:")
report_first_classifier_performance(mlp_ensemble, X, y)
```

```
MLPClassifier Ensemble:
Accuracy of the first classifier in the ensemble: 0.55
```

[80]:
```
ensemble_fit(dtc_ensemble, X, y)

print("\nDecisionTreeClassifier Ensemble:")
report_first_classifier_performance(dtc_ensemble, X, y)
```

```
DecisionTreeClassifier Ensemble:
Accuracy of the first classifier in the ensemble: 0.85
```

0.4 Problem 4

[118]:
```
ensembles = {
    "GaussianNB Ensemble": gnb_ensemble,
    "SVC Ensemble": svc_ensemble,
    "MLPClassifier Ensemble": mlp_ensemble,
    "DecisionTreeClassifier Ensemble": dtc_ensemble
}
for name, ensemble in ensembles.items():
    print(f"\nFitting {name}...")
    ensemble_fit(ensemble, X, y)
    print(f"\n{name} Performance:")
    report_first_classifier_performance(ensemble, X, y)


def ensemble_predict(ensemble, X):
    probas = np.zeros((X.shape[0], 2))
    for clf in ensemble:
        probas += clf.predict_proba(X)
    avg_probas = probas / len(ensemble)
    final_predictions = np.argmax(avg_probas, axis=1)
    return final_predictions
```

```
Fitting GaussianNB Ensemble…

GaussianNB Ensemble Performance:
Accuracy of the first classifier in the ensemble: 0.86

Fitting SVC Ensemble…
```

```
SVC Ensemble Performance:
Accuracy of the first classifier in the ensemble: 0.59

Fitting MLPClassifier Ensemble…

MLPClassifier Ensemble Performance:
Accuracy of the first classifier in the ensemble: 0.50

Fitting DecisionTreeClassifier Ensemble…

DecisionTreeClassifier Ensemble Performance:
Accuracy of the first classifier in the ensemble: 0.87
```

[119]:
```
sample_input = X[:1]

gnb_predictions = ensemble_predict(gnb_ensemble, sample_input)
print("GaussianNB Ensemble Prediction:", gnb_predictions)
```

```
GaussianNB Ensemble Prediction: [0]
```

[120]:
```
sample_input = X[:1]

gnb_predictions = ensemble_predict(gnb_ensemble, sample_input)
svc_predictions = ensemble_predict(svc_ensemble, sample_input)
mlp_predictions = ensemble_predict(mlp_ensemble, sample_input)
dtc_predictions = ensemble_predict(dtc_ensemble, sample_input)

print("GaussianNB Ensemble Prediction:", gnb_predictions)
print("SVC Ensemble Prediction:", svc_predictions)
print("MLPClassifier Ensemble Prediction:", mlp_predictions)
print("DecisionTreeClassifier Ensemble Prediction:", dtc_predictions)
```

```
GaussianNB Ensemble Prediction: [0]
SVC Ensemble Prediction: [1]
MLPClassifier Ensemble Prediction: [1]
DecisionTreeClassifier Ensemble Prediction: [0]
```

0.5 Problem 5

[143]:
```
def ensemble_fit_with_subsample(ensemble, X, y, subsample_ratio):
    for clf in ensemble:
        while True:
            X_sample, y_sample = resample(X, y, replace=True,␣
 ↪n_samples=int(subsample_ratio * len(X)))
            if len(np.unique(y_sample)) > 1:
                break
        clf.fit(X_sample, y_sample)
```

```
def evaluate_ensemble(ensemble, X, y, subsample_ratio, cv_folds=10):
    kf = StratifiedKFold(n_splits=cv_folds, shuffle=True, random_state=1)
    scores = []

    for train_index, test_index in kf.split(X, y):
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]

        ensemble_fit_with_subsample(ensemble, X_train, y_train, subsample_ratio)

        predictions = ensemble_predict(ensemble, X_test)

        accuracy = accuracy_score(y_test, predictions)
        scores.append(accuracy)

    return np.array(scores)

for name, ensemble in ensembles.items():
    scores = evaluate_ensemble(ensemble, X, y, subsample_ratio=0.2, cv_folds=10)
    print(f"{name} 10-fold CV Mean Accuracy: {np.mean(scores):.4f} ± {np.
 ↪std(scores):.4f}")
```

```
GaussianNB Ensemble 10-fold CV Mean Accuracy: 0.8595 ± 0.0383
SVC Ensemble 10-fold CV Mean Accuracy: 0.5196 ± 0.0430
MLPClassifier Ensemble 10-fold CV Mean Accuracy: 0.5284 ± 0.0292
DecisionTreeClassifier Ensemble 10-fold CV Mean Accuracy: 0.8486 ± 0.0521
```

[126]:
```
def evaluate_decision_tree(X, y, subsample_ratio, cv_folds=10):

    kf = StratifiedKFold(n_splits=cv_folds, shuffle=True, random_state=1)
    scores = []
    for train_index, test_index in kf.split(X, y):
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]

        X_sample, y_sample = resample(X_train, y_train, replace=True,␣
 ↪n_samples=int(subsample_ratio * len(X_train)))

        clf = DecisionTreeClassifier(max_depth=5, max_features=5)
        clf.fit(X_sample, y_sample)

        y_pred = clf.predict(X_test)

        accuracy = accuracy_score(y_test, y_pred)
        scores.append(accuracy)

    return np.array(scores)
```

```
dt_scores_005 = evaluate_decision_tree(X, y, subsample_ratio=0.05, cv_folds=10)
print(f"Regular Decision Tree 10-fold CV Mean Accuracy with subsample ratio 0.
 ↪05: {np.mean(dt_scores_005):.4f} ± {np.std(dt_scores_005):.4f}")
```

```
Regular Decision Tree 10-fold CV Mean Accuracy with subsample ratio 0.05: 0.7614
± 0.0371
```

0.6 Problem 6

[149]:
```
def create_ensembles():
    gnb_ensemble = [GaussianNB() for _ in range(10)]
    svc_ensemble = [SVC(kernel='linear', probability=True, max_iter=30) for _␣
 ↪in range(10)]
    mlp_ensemble = [MLPClassifier(hidden_layer_sizes=(3, 3), max_iter=30,␣
 ↪tol=1e-1) for _ in range(10)]
    dtc_ensemble = [DecisionTreeClassifier(max_depth=5, max_features=5) for _␣
 ↪in range(10)]
    return {
        "GaussianNB Ensemble": gnb_ensemble,
        "SVC Ensemble": svc_ensemble,
        "MLPClassifier Ensemble": mlp_ensemble,
        "DecisionTreeClassifier Ensemble": dtc_ensemble
    }


def evaluate_single_classifier(clf, X, y, subsample_ratio, cv_folds=10):
    kf = StratifiedKFold(n_splits=cv_folds, shuffle=True, random_state=1)
    scores = []

    for train_index, test_index in kf.split(X, y):
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]

        while True:
            X_sample, y_sample = resample(X_train, y_train, replace=True,␣
 ↪n_samples=int(subsample_ratio * len(X_train)))
            if len(np.unique(y_sample)) > 1:
                break

        clf.fit(X_sample, y_sample)

        y_pred = clf.predict(X_test)
        accuracy = accuracy_score(y_test, y_pred)
        scores.append(accuracy)
```

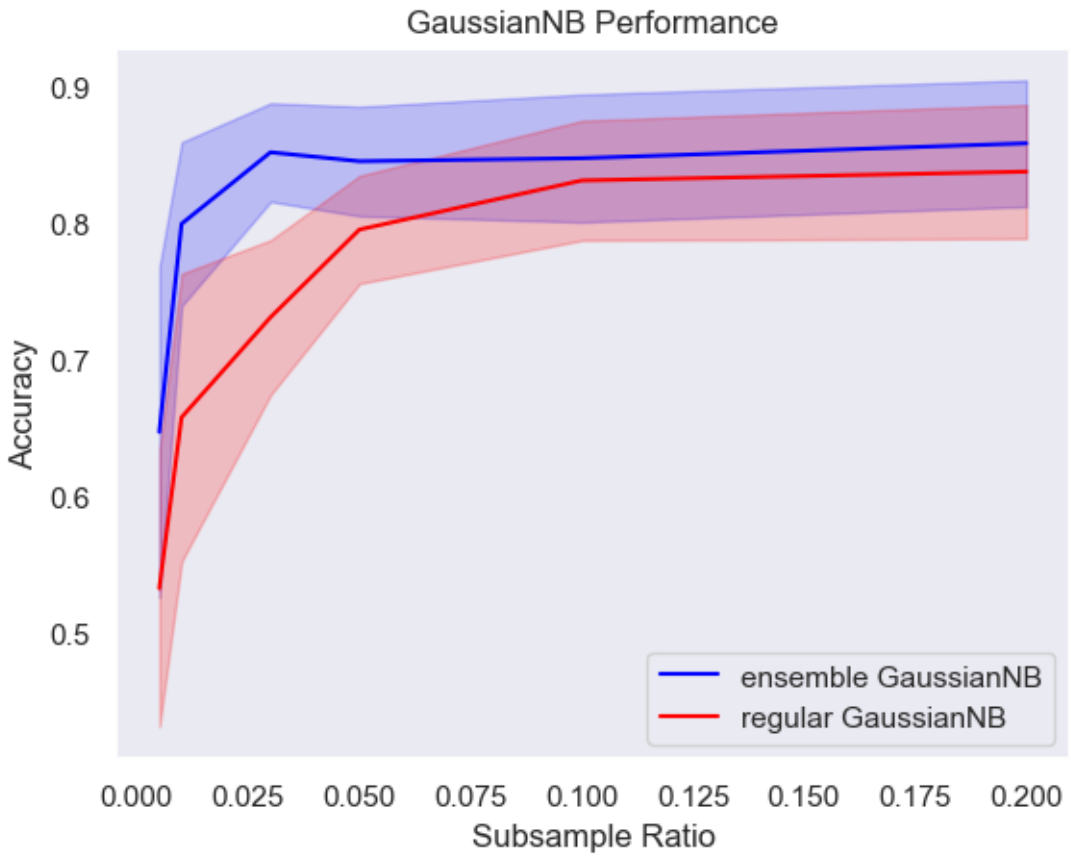
```
    return np.array(scores)


subsample_ratios = [0.005, 0.01, 0.03, 0.05, 0.1, 0.2]

results = {
    "GaussianNB": {"ensemble": [], "regular": []},
    "SVC": {"ensemble": [], "regular": []},
    "MLPClassifier": {"ensemble": [], "regular": []},
    "DecisionTreeClassifier": {"ensemble": [], "regular": []}
}

# Evaluate ensembles and regular classifiers for each subsample ratio
for subsample_ratio in subsample_ratios:
    print(f"\nEvaluating with subsample ratio of {subsample_ratio}:")

    # Create fresh ensembles for each subsample ratio
    ensembles = create_ensembles()

    for name, ensemble in ensembles.items():
        scores = evaluate_ensemble(ensemble, X, y, subsample_ratio, cv_folds=10)
        classifier_type = name.split()[0]
        results[classifier_type]["ensemble"].append((subsample_ratio, np.
 ↪mean(scores), np.std(scores)))
        print(f"{name} 10-fold CV Mean Accuracy: {np.mean(scores):.4f} ± {np.
 ↪std(scores):.4f}")

    # Evaluate regular classifiers
    regular_classifiers = {
        "GaussianNB": GaussianNB(),
        "SVC": SVC(kernel='linear', probability=True, max_iter=30),
        "MLPClassifier": MLPClassifier(hidden_layer_sizes=(3, 3), max_iter=30,␣
 ↪tol=1e-1),
        "DecisionTreeClassifier": DecisionTreeClassifier(max_depth=5,␣
 ↪max_features=5)
    }

    for name, clf in regular_classifiers.items():
        scores = evaluate_single_classifier(clf, X, y, subsample_ratio,␣
 ↪cv_folds=10)
        classifier_type = name
        results[classifier_type]["regular"].append((subsample_ratio, np.
 ↪mean(scores), np.std(scores)))
        print(f"{name} with subsample ratio {subsample_ratio} 10-fold CV Mean␣
 ↪Accuracy: {np.mean(scores):.4f} ± {np.std(scores):.4f}")
```

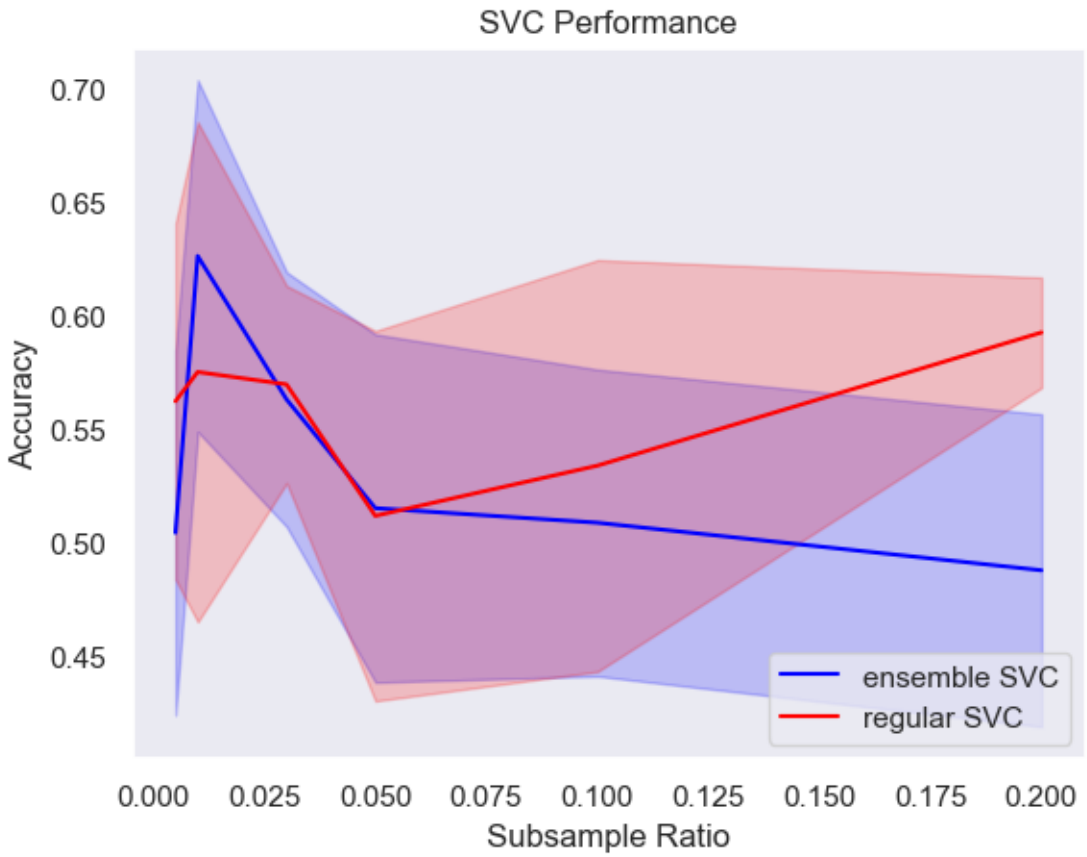
```
Evaluating with subsample ratio of 0.005:
GaussianNB Ensemble 10-fold CV Mean Accuracy: 0.6470 ± 0.1210
SVC Ensemble 10-fold CV Mean Accuracy: 0.5044 ± 0.0803
MLPClassifier Ensemble 10-fold CV Mean Accuracy: 0.4945 ± 0.0449
DecisionTreeClassifier Ensemble 10-fold CV Mean Accuracy: 0.7627 ± 0.0838
GaussianNB with subsample ratio 0.005 10-fold CV Mean Accuracy: 0.5325 ± 0.1012
SVC with subsample ratio 0.005 10-fold CV Mean Accuracy: 0.5623 ± 0.0785
MLPClassifier with subsample ratio 0.005 10-fold CV Mean Accuracy: 0.5263 ±
0.0654
DecisionTreeClassifier with subsample ratio 0.005 10-fold CV Mean Accuracy:
0.6395 ± 0.1228

Evaluating with subsample ratio of 0.01:
GaussianNB Ensemble 10-fold CV Mean Accuracy: 0.7995 ± 0.0599
SVC Ensemble 10-fold CV Mean Accuracy: 0.6263 ± 0.0774
MLPClassifier Ensemble 10-fold CV Mean Accuracy: 0.5350 ± 0.0742
DecisionTreeClassifier Ensemble 10-fold CV Mean Accuracy: 0.8203 ± 0.0363
GaussianNB with subsample ratio 0.01 10-fold CV Mean Accuracy: 0.6577 ± 0.1053
SVC with subsample ratio 0.01 10-fold CV Mean Accuracy: 0.5752 ± 0.1100
MLPClassifier with subsample ratio 0.01 10-fold CV Mean Accuracy: 0.5097 ±
0.0716
DecisionTreeClassifier with subsample ratio 0.01 10-fold CV Mean Accuracy:
0.6675 ± 0.1351

Evaluating with subsample ratio of 0.03:
GaussianNB Ensemble 10-fold CV Mean Accuracy: 0.8519 ± 0.0360
SVC Ensemble 10-fold CV Mean Accuracy: 0.5631 ± 0.0560
MLPClassifier Ensemble 10-fold CV Mean Accuracy: 0.4935 ± 0.0563
DecisionTreeClassifier Ensemble 10-fold CV Mean Accuracy: 0.8258 ± 0.0340
GaussianNB with subsample ratio 0.03 10-fold CV Mean Accuracy: 0.7310 ± 0.0567
SVC with subsample ratio 0.03 10-fold CV Mean Accuracy: 0.5697 ± 0.0434
MLPClassifier with subsample ratio 0.03 10-fold CV Mean Accuracy: 0.5240 ±
0.0628
DecisionTreeClassifier with subsample ratio 0.03 10-fold CV Mean Accuracy:
0.7082 ± 0.0784

Evaluating with subsample ratio of 0.05:
GaussianNB Ensemble 10-fold CV Mean Accuracy: 0.8453 ± 0.0400
SVC Ensemble 10-fold CV Mean Accuracy: 0.5152 ± 0.0766
MLPClassifier Ensemble 10-fold CV Mean Accuracy: 0.5098 ± 0.0741
DecisionTreeClassifier Ensemble 10-fold CV Mean Accuracy: 0.8290 ± 0.0364
GaussianNB with subsample ratio 0.05 10-fold CV Mean Accuracy: 0.7953 ± 0.0395
SVC with subsample ratio 0.05 10-fold CV Mean Accuracy: 0.5118 ± 0.0815
MLPClassifier with subsample ratio 0.05 10-fold CV Mean Accuracy: 0.4924 ±
0.0502
DecisionTreeClassifier with subsample ratio 0.05 10-fold CV Mean Accuracy:
0.7616 ± 0.0651
```

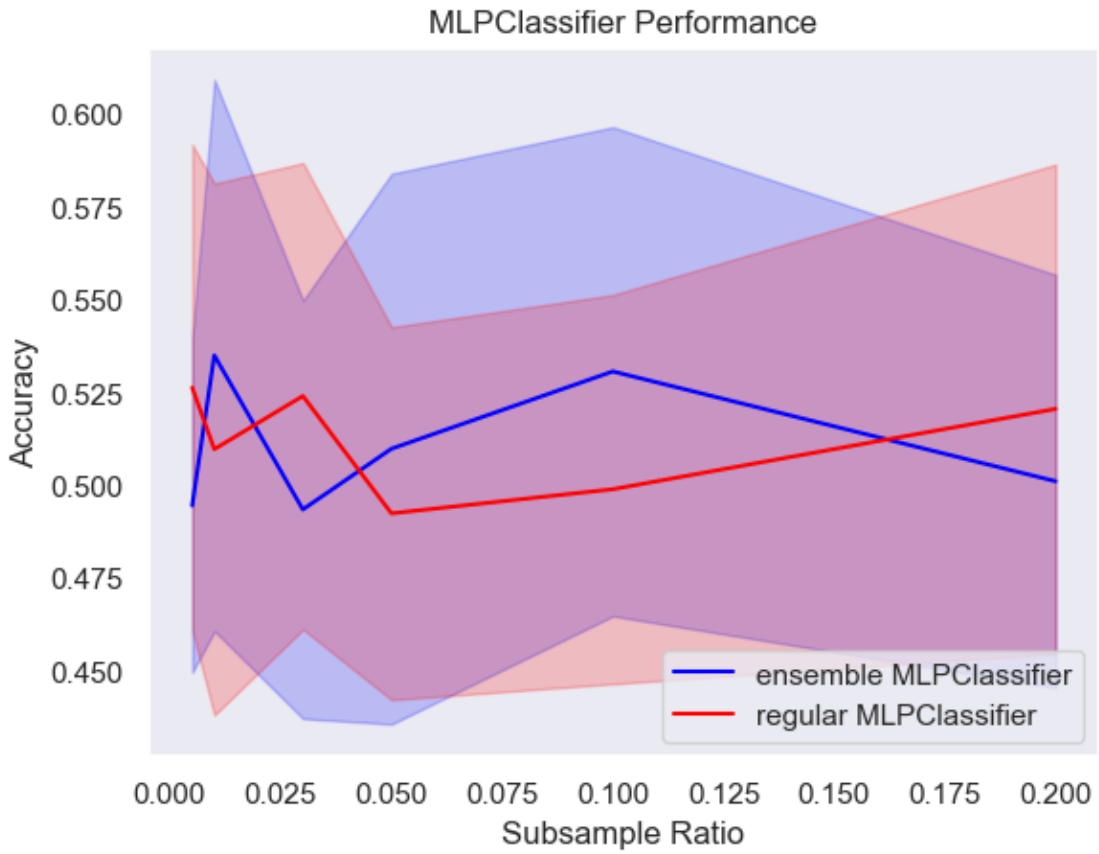
```
Evaluating with subsample ratio of 0.1:
GaussianNB Ensemble 10-fold CV Mean Accuracy: 0.8475 ± 0.0468
SVC Ensemble 10-fold CV Mean Accuracy: 0.5088 ± 0.0675
MLPClassifier Ensemble 10-fold CV Mean Accuracy: 0.5306 ± 0.0659
DecisionTreeClassifier Ensemble 10-fold CV Mean Accuracy: 0.8399 ± 0.0460
GaussianNB with subsample ratio 0.1 10-fold CV Mean Accuracy: 0.8312 ± 0.0438
SVC with subsample ratio 0.1 10-fold CV Mean Accuracy: 0.5339 ± 0.0906
MLPClassifier with subsample ratio 0.1 10-fold CV Mean Accuracy: 0.4989 ± 0.0524
DecisionTreeClassifier with subsample ratio 0.1 10-fold CV Mean Accuracy: 0.7408
± 0.0508

Evaluating with subsample ratio of 0.2:
GaussianNB Ensemble 10-fold CV Mean Accuracy: 0.8584 ± 0.0462
SVC Ensemble 10-fold CV Mean Accuracy: 0.4879 ± 0.0688
MLPClassifier Ensemble 10-fold CV Mean Accuracy: 0.5011 ± 0.0556
DecisionTreeClassifier Ensemble 10-fold CV Mean Accuracy: 0.8584 ± 0.0458
GaussianNB with subsample ratio 0.2 10-fold CV Mean Accuracy: 0.8377 ± 0.0489
SVC with subsample ratio 0.2 10-fold CV Mean Accuracy: 0.5926 ± 0.0242
MLPClassifier with subsample ratio 0.2 10-fold CV Mean Accuracy: 0.5205 ± 0.0659
DecisionTreeClassifier with subsample ratio 0.2 10-fold CV Mean Accuracy: 0.7855
± 0.0537
```

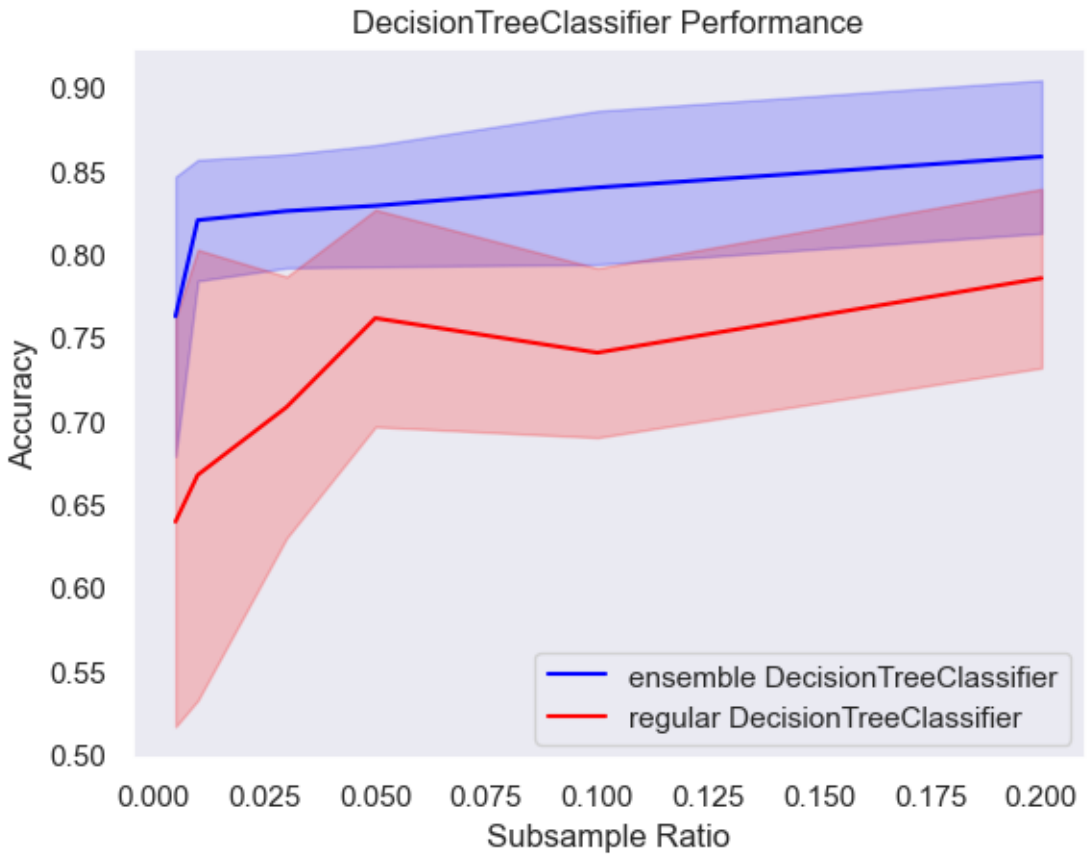
0.7 Problem 7

[152]:
```
colors = {
    "ensemble": "blue",
    "regular": "red"
}

for classifier_type, data in results.items():
    plt.figure()
    for label, color in colors.items():
        vals, accs, stdevs = zip(*data[label])

        vals, accs, stdevs = np.array(vals), np.array(accs), np.array(stdevs)

        plt.plot(vals, accs, label=f'{label} {classifier_type}', color=color)
        plt.fill_between(vals, accs - stdevs, accs + stdevs, color=color,␣
 ↪alpha=0.2)

    plt.xlabel('Subsample Ratio')
    plt.ylabel('Accuracy')
    plt.legend(loc='lower right')
    plt.grid()
    plt.title(f'{classifier_type} Performance')
    plt.show()
```


GaussianNB Performance
0.9
0.8
0.7
0.6
0.5
Accuracy
ensemble GaussianNB
regular GaussianNB
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
Subsample Ratio


SVC Performance
0.70
0.65
0.60
0.55
0.50
0.45
Accuracy
ensemble SVC
regular SVC
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
Subsample Ratio


MLPClassifier Performance
Accuracy
0.600
0.575
0.550
0.525
0.500
0.475
0.450
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
Subsample Ratio
ensemble MLPClassifier
regular MLPClassifier


DecisionTreeClassifier Performance
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
Accuracy
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
Subsample Ratio
ensemble DecisionTreeClassifier
regular DecisionTreeClassifier

[ ]: