example

January 23, 2024

[9]:
```
import matplotlib.pyplot as plt
from ucimlrepo import fetch_ucirepo
import pandas as pd
import numpy as np
```

[10]:
```
mushroom = fetch_ucirepo(id=73)
mushroom_features_df = pd.DataFrame.from_dict(mushroom['data']['features'])
mushroom_classes_df = pd.DataFrame.from_dict(mushroom['data']['targets'])
pd.concat([mushroom_classes_df, mushroom_features_df], axis=1)
```

[10]:

	poisonous	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	…
0	p	x	s	n	t	p	f	
1	e	x	s	y	t	a	f	
2	e	b	s	w	t	l	f	
3	p	x	y	w	t	p	f	
4	e	x	s	g	f	n	f	
…	…	…	…	…	…	…	…	
8119	e	k	s	n	f	n	a	
8120	e	x	s	n	f	n	a	
8121	e	f	s	n	f	n	a	
8122	p	k	y	n	f	y	f	
8123	e	x	s	n	f	n	a	

	gill-spacing	gill-size	gill-color	…	stalk-surface-below-ring	…
0	c	n	k	…	s	
1	c	b	k	…	s	
2	c	b	n	…	s	
3	c	n	n	…	s	
4	w	b	k	…	s	
…	…	…	…	…	…	
8119	c	b	y	…	s	
8120	c	b	y	…	s	
8121	c	b	n	…	s	
8122	c	n	b	…	k	
8123	c	b	y	…	s	

	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	…
0	w	w	p	w	

```
1                        w                      w         p          w
2                        w                      w         p          w
3                        w                      w         p          w
4                        w                      w         p          w
…                      …                      …       …          …
8119                     o                      o         p          o
8120                     o                      o         p          n
8121                     o                      o         p          o
8122                     w                      w         p          w
8123                     o                      o         p          o

     ring-number ring-type spore-print-color population habitat
0              o         p                 k          s       u
1              o         p                 n          n       g
2              o         p                 n          n       m
3              o         p                 k          s       u
4              o         e                 n          a       g
…            …       …                 …          …     …
8119           o         p                 b          c       l
8120           o         p                 b          v       l
8121           o         p                 b          c       l
8122           o         e                 w          v       l
8123           o         p                 o          c       l

[8124 rows x 23 columns]
```

[11]:
```
breast_cancer = fetch_ucirepo(id=17)
breast_cancer_features_df = pd.DataFrame.
 ↪from_dict(breast_cancer['data']['features'])
breast_cancer_classes_df = pd.DataFrame.
 ↪from_dict(breast_cancer['data']['targets'])
pd.concat([breast_cancer_classes_df, breast_cancer_features_df], axis=1)
```

[11]:
```
    Diagnosis  radius1  texture1  perimeter1   area1  smoothness1  \
0           M    17.99     10.38      122.80  1001.0      0.11840
1           M    20.57     17.77      132.90  1326.0      0.08474
2           M    19.69     21.25      130.00  1203.0      0.10960
3           M    11.42     20.38       77.58   386.1      0.14250
4           M    20.29     14.34      135.10  1297.0      0.10030
..        …      …         …          …       …           …
564         M    21.56     22.39      142.00  1479.0      0.11100
565         M    20.13     28.25      131.20  1261.0      0.09780
566         M    16.60     28.08      108.30   858.1      0.08455
567         M    20.60     29.33      140.10  1265.0      0.11780
568         B     7.76     24.54       47.92   181.0      0.05263

     compactness1  concavity1  concave_points1  symmetry1  …  radius3  \
```

```
0         0.27760     0.30010          0.14710     0.2419  …   25.380
1         0.07864     0.08690          0.07017     0.1812  …   24.990
2         0.15990     0.19740          0.12790     0.2069  …   23.570
3         0.28390     0.24140          0.10520     0.2597  …   14.910
4         0.13280     0.19800          0.10430     0.1809  …   22.540
..            …           …                …       …   …       …
564       0.11590     0.24390          0.13890     0.1726  …   25.450
565       0.10340     0.14400          0.09791     0.1752  …   23.690
566       0.10230     0.09251          0.05302     0.1590  …   18.980
567       0.27700     0.35140          0.15200     0.2397  …   25.740
568       0.04362     0.00000          0.00000     0.1587  …    9.456

     texture3  perimeter3   area3  smoothness3  compactness3  concavity3  \
0       17.33      184.60  2019.0      0.16220       0.66560      0.7119
1       23.41      158.80  1956.0      0.12380       0.18660      0.2416
2       25.53      152.50  1709.0      0.14440       0.42450      0.4504
3       26.50       98.87   567.7      0.20980       0.86630      0.6869
4       16.67      152.20  1575.0      0.13740       0.20500      0.4000
..        …          …     …           …             …            …
564     26.40      166.10  2027.0      0.14100       0.21130      0.4107
565     38.25      155.00  1731.0      0.11660       0.19220      0.3215
566     34.12      126.70  1124.0      0.11390       0.30940      0.3403
567     39.42      184.60  1821.0      0.16500       0.86810      0.9387
568     30.37       59.16   268.6      0.08996       0.06444      0.0000

     concave_points3  symmetry3  fractal_dimension3
0             0.2654     0.4601             0.11890
1             0.1860     0.2750             0.08902
2             0.2430     0.3613             0.08758
3             0.2575     0.6638             0.17300
4             0.1625     0.2364             0.07678
..               …          …                …
564           0.2216     0.2060             0.07115
565           0.1628     0.2572             0.06637
566           0.1418     0.2218             0.07820
567           0.2650     0.4087             0.12400
568           0.0000     0.2871             0.07039

[569 rows x 31 columns]
```

[12]:
```
from sklearn.preprocessing import LabelEncoder, LabelBinarizer
from sklearn.datasets import fetch_openml

titanic = fetch_openml(name='titanic', version=1)
titanic_df = titanic['frame']
titanic_df['boat'] = titanic_df['boat'].fillna(value=0)
titanic_df['body'] = titanic_df['body'].fillna(value=0)
```

```
titanic_df['body'] = titanic_df['body'].fillna(value=0)

titanic_df
```

[12]:

```
      pclass survived                                             name  \
0          1        1                    Allen, Miss. Elisabeth Walton
1          1        1                   Allison, Master. Hudson Trevor
2          1        0                     Allison, Miss. Helen Loraine
3          1        0             Allison, Mr. Hudson Joshua Creighton
4          1        0  Allison, Mrs. Hudson J C (Bessie Waldo Daniels)
…        …      …                                                 …
1304       3        0                          Zabour, Miss. Hileni
1305       3        0                         Zabour, Miss. Thamine
1306       3        0                     Zakarian, Mr. Mapriededer
1307       3        0                           Zakarian, Mr. Ortin
1308       3        0                            Zimmerman, Mr. Leo

         sex      age  sibsp  parch  ticket      fare    cabin embarked boat  \
0     female  29.0000      0      0   24160  211.3375       B5        S    2
1       male   0.9167      1      2  113781  151.5500  C22 C26        S   11
2     female   2.0000      1      2  113781  151.5500  C22 C26        S    0
3       male  30.0000      1      2  113781  151.5500  C22 C26        S    0
4     female  25.0000      1      2  113781  151.5500  C22 C26        S    0
…       …       …    …    …    …      …       …       …  …
1304  female  14.5000      1      0    2665   14.4542      NaN        C    0
1305  female      NaN      1      0    2665   14.4542      NaN        C    0
1306    male  26.5000      0      0    2656    7.2250      NaN        C    0
1307    male  27.0000      0      0    2670    7.2250      NaN        C    0
1308    male  29.0000      0      0  315082    7.8750      NaN        S    0

       body                        home.dest
0       0.0                     St Louis, MO
1       0.0  Montreal, PQ / Chesterville, ON
2       0.0  Montreal, PQ / Chesterville, ON
3     135.0  Montreal, PQ / Chesterville, ON
4       0.0  Montreal, PQ / Chesterville, ON
…     …                                   …
1304  328.0                             NaN
1305    0.0                             NaN
1306  304.0                             NaN
1307    0.0                             NaN
1308    0.0                             NaN

[1309 rows x 14 columns]
```

[13]:
```
titanic_df_encoded = titanic_df

titanic_df['boat'] = titanic_df['boat'].astype(str)
titanic_df['boat'] = titanic_df['boat'].astype(str)

encoded_features = ['sex', 'cabin', 'ticket', 'home.dest', 'embarked', 'boat']
label_encoders = {feature: LabelEncoder() for feature in encoded_features}
for feature, encoder in label_encoders.items():
    titanic_df_encoded[feature] = encoder.fit_transform(titanic_df[feature])

titanic_lb_survived = LabelBinarizer()
titanic_df_encoded['survived'] = titanic_lb_survived.
 ↪fit_transform(titanic_df['survived'])

titanic_df_encoded
```

[13]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1	1	Allen, Miss. Elisabeth Walton	0	29.0000	0	0	187	211.3375	43	2	12	0.0	308
1	1	1	Allison, Master. Hudson Trevor	1	0.9167	1	2	49	151.5500	79	2	3	0.0	230
2	1	0	Allison, Miss. Helen Loraine	0	2.0000	1	2	49	151.5500	79	2	0	0.0	230
3	1	0	Allison, Mr. Hudson Joshua Creighton	1	30.0000	1	2	49	151.5500	79	2	0	135.0	230
4	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	0	25.0000	1	2	49	151.5500	79	2	0	0.0	230
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
1304	3	0	Zabour, Miss. Hileni	0	14.5000	1	0	259	14.4542	186	0	0	328.0	308
1305	3	0	Zabour, Miss. Thamine	0	NaN	1	0	259	14.4542	186	0	0	0.0	308
1306	3	0	Zakarian, Mr. Mapriededer	1	26.5000	0	0	250	7.2250	186	0	0	304.0	308
1307	3	0	Zakarian, Mr. Ortin	1	27.0000	0	0	264	7.2250	186	0	0	0.0	308
1308	3	0	Zimmerman, Mr. Leo	1	29.0000	0	0	346	7.8750	186	2	0	0.0	308

```
4            230
…            …
1304         369
1305         369
1306         369
1307         369
1308         369

[1309 rows x 14 columns]
```

0.0.1 Titanic example

[14]:
```
def evaluate(clf, X, y, scoring=None) -> dict:
    scoring = scoring or ['accuracy', 'precision', 'recall', 'roc_auc', 'f1']

    cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=0)
    _scores = cross_validate(clf, X, y, cv=cv, scoring=scoring, n_jobs=8,␣
 ↪return_train_score=True)
    return {name: score.mean() for name, score in _scores.items()}


def evaluate_return_estimator(clf, X, y, scoring=None) -> tuple[dict, list]:
    scoring = scoring or ['accuracy', 'precision', 'recall', 'roc_auc', 'f1']

    cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=0)
    _scores = cross_validate(clf, X, y, cv=cv, scoring=scoring, n_jobs=8,␣
 ↪return_train_score=True,
                             return_estimator=True)
    return {name: score.mean() for name, score in _scores.items() if name !=␣
 ↪'estimator'}, _scores['estimator']
```

[15]:
```
from sklearn.ensemble import RandomForestClassifier
```

[16]:
```
from sklearn.model_selection import cross_validate, ShuffleSplit

N_ESTIMATORS = 200

clf = RandomForestClassifier(n_estimators=N_ESTIMATORS)

X = titanic_df_encoded.drop(['survived', 'name'], axis='columns')
y = titanic_df['survived']

evaluate(clf, X, y, scoring=['accuracy', 'precision', 'recall', 'roc_auc',␣
 ↪'f1'])
```

[16]:
```
{'fit_time': 0.31610894203186035,
 'score_time': 0.030126237869262697,
```

```
 'test_accuracy': 0.9702290076335878,
 'train_accuracy': 0.9793696275071634,
 'test_precision': 0.9787358059706378,
 'train_precision': 0.9821955063295645,
 'test_recall': 0.9408521106374348,
 'train_recall': 0.9635912785219982,
 'test_roc_auc': 0.9932495116763438,
 'train_roc_auc': 0.9982890332080114,
 'test_f1': 0.9594007649305427,
 'train_f1': 0.9728011290549146}
```

0.0.2 Breast cancer example

[17]:
```
clf = RandomForestClassifier(n_estimators=N_ESTIMATORS)

X = breast_cancer_features_df
breast_cancer_lb_diagnosis = LabelBinarizer()

X = breast_cancer_features_df.to_numpy()
y = np.squeeze(breast_cancer_lb_diagnosis.
 ↪fit_transform(breast_cancer_classes_df))

evaluate(clf, X, y, scoring=['accuracy', 'precision', 'recall', 'roc_auc',␣
 ↪'f1'])
```

[17]:
```
{'fit_time': 0.3469456672668457,
 'score_time': 0.02559514045715332,
 'test_accuracy': 0.9631578947368421,
 'train_accuracy': 1.0,
 'test_precision': 0.9698643410852712,
 'train_precision': 1.0,
 'test_recall': 0.936071449815499,
 'train_recall': 1.0,
 'test_roc_auc': 0.9909137176413665,
 'train_roc_auc': 1.0,
 'test_f1': 0.9516461901101048,
 'train_f1': 1.0}
```

0.0.3 Mushroom example

[63]:
```
clf = RandomForestClassifier(n_estimators=N_ESTIMATORS)

mushroom_lb_poisonous = LabelBinarizer()

mushroom_features_df_encoded = mushroom_features_df

encoded_features = mushroom_features_df.columns
```

```
label_encoders = {feature: LabelEncoder() for feature in encoded_features}
for feature, encoder in label_encoders.items():
    mushroom_features_df[feature] = encoder.
 ↪fit_transform(mushroom_features_df[feature])

X = mushroom_features_df_encoded
y = np.squeeze(mushroom_lb_poisonous.fit_transform(mushroom_classes_df))

evaluate(clf, X, y, scoring=['accuracy', 'precision', 'recall', 'roc_auc',␣
 ↪'f1'])
```

[63]:
```
{'fit_time': 0.4403873920440674,
 'score_time': 0.046620416641235354,
 'test_accuracy': 1.0,
 'train_accuracy': 1.0,
 'test_precision': 1.0,
 'train_precision': 1.0,
 'test_recall': 1.0,
 'train_recall': 1.0,
 'test_roc_auc': 1.0,
 'train_roc_auc': 1.0,
 'test_f1': 1.0,
 'train_f1': 1.0}
```

1 Boosted RF

Plain ol' AdaBoost

[69]:
```
from sklearn.ensemble import AdaBoostClassifier

clf = AdaBoostClassifier(n_estimators=N_ESTIMATORS, algorithm='SAMME')

X = breast_cancer_features_df
breast_cancer_lb_diagnosis = LabelBinarizer()

y = np.squeeze(breast_cancer_lb_diagnosis.
 ↪fit_transform(breast_cancer_classes_df))

evaluate(clf, X, y, scoring=['accuracy', 'precision', 'recall', 'roc_auc',␣
 ↪'f1'])
```

[69]:
```
{'fit_time': 0.5341531753540039,
 'score_time': 0.052882146835327146,
 'test_accuracy': 0.9719298245614034,
 'train_accuracy': 1.0,
 'test_precision': 0.9769461694209735,
 'train_precision': 1.0,
```

```
 'test_recall': 0.9497502722903775,
 'train_recall': 1.0,
 'test_roc_auc': 0.9966182602092898,
 'train_roc_auc': 1.0,
 'test_f1': 0.9629576928742024,
 'train_f1': 1.0}
```

"Boosted RF"

[20]:
```
from IPython.display import display
from sklearn.tree import DecisionTreeClassifier

BOOST_RF_N_ESTIMATORS = 50

clf_ref = AdaBoostClassifier(n_estimators=BOOST_RF_N_ESTIMATORS,␣
 ↪algorithm='SAMME')


class BoostedRFClassifier(AdaBoostClassifier):
    def __init__(
            self,
            *,
            estimator=DecisionTreeClassifier(),
            n_estimators=BOOST_RF_N_ESTIMATORS,
            learning_rate=1.0,
            random_state=None,
            algorithm='SAMME',
            _disable_weights=False
    ):
        super().__init__(
            estimator=estimator,
            n_estimators=n_estimators,
            learning_rate=learning_rate,
            random_state=random_state,
            algorithm=algorithm
        )

        self._disable_weights = _disable_weights
        self._estimator_weights = np.zeros(self.n_estimators, dtype=np.float64)

    @property
    def estimator_weights_(self):
        return np.ones(self.n_estimators, dtype=np.float64) if self.
 ↪_disable_weights else self._estimator_weights

    @estimator_weights_.setter
    def estimator_weights_(self, v):
```

```
        if not self._disable_weights:
            self._estimator_weights = v


clf = BoostedRFClassifier(
    estimator=DecisionTreeClassifier(max_depth=5),
    n_estimators=BOOST_RF_N_ESTIMATORS,
    algorithm='SAMME',
    _disable_weights=True
)

X = breast_cancer_features_df
breast_cancer_lb_diagnosis = LabelBinarizer()

y = np.squeeze(breast_cancer_lb_diagnosis.
 ↪fit_transform(breast_cancer_classes_df))

res_ref = evaluate(clf_ref, X, y, scoring=['accuracy', 'precision', 'recall',␣
 ↪'roc_auc', 'f1'])
res, est = evaluate_return_estimator(clf, X, y, scoring=['accuracy',␣
 ↪'precision', 'recall', 'roc_auc', 'f1'])

display(
    pd.concat([
        pd.DataFrame.from_dict(res_ref, orient='index')[0].rename('AdaBoost'),
        pd.DataFrame.from_dict(res, orient='index')[0].rename('Boosted RF')
    ], axis=1)
)
```

	AdaBoost	Boosted RF
fit_time	0.144744	0.131094
score_time	0.020160	0.011134
test_accuracy	0.973684	0.961404
train_accuracy	1.000000	1.000000
test_precision	0.982013	0.972925
train_precision	1.000000	1.000000
test_recall	0.950175	0.927043
train_recall	1.000000	1.000000
test_roc_auc	0.994809	0.993863
train_roc_auc	1.000000	1.000000
test_f1	0.965594	0.948692
train_f1	1.000000	1.000000

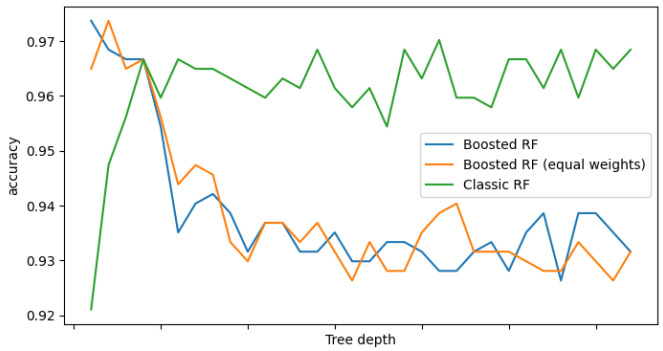
[21]:
```
tree_depths = list(range(1, 33))
measures = ['accuracy', 'precision', 'recall', 'roc_auc', 'f1']
res = [evaluate(BoostedRFClassifier(
    estimator=DecisionTreeClassifier(max_depth=depth),
```

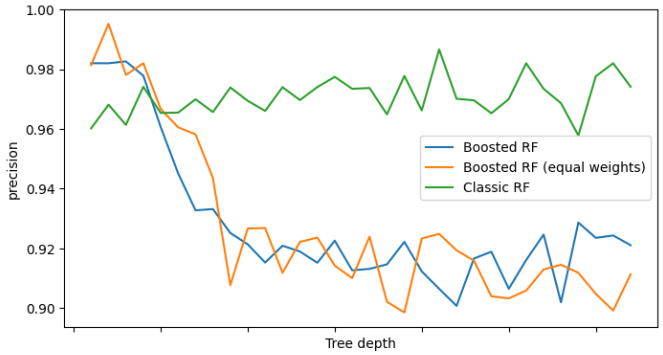
```
    n_estimators=BOOST_RF_N_ESTIMATORS,
    algorithm='SAMME',
    _disable_weights=False
), X, y, scoring=measures) for depth in tree_depths]
```

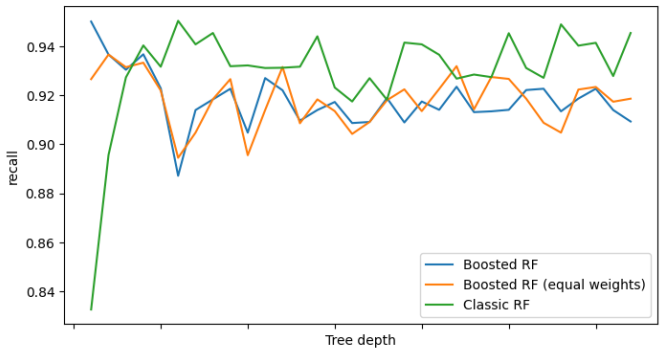
[22]:
```
res_no_weights = [evaluate(BoostedRFClassifier(
    estimator=DecisionTreeClassifier(max_depth=depth),
    n_estimators=BOOST_RF_N_ESTIMATORS,
    algorithm='SAMME',
    _disable_weights=True
), X, y, scoring=measures) for depth in tree_depths]
```

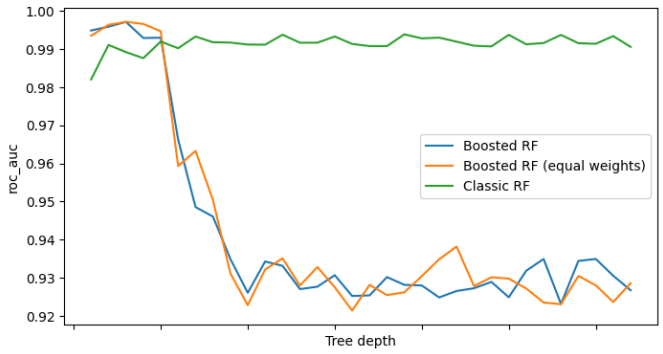
[23]:
```
res_ref = [evaluate(RandomForestClassifier(
    max_depth=depth,
    n_estimators=BOOST_RF_N_ESTIMATORS,
    n_jobs=8
), X, y, scoring=measures) for depth in tree_depths]
```

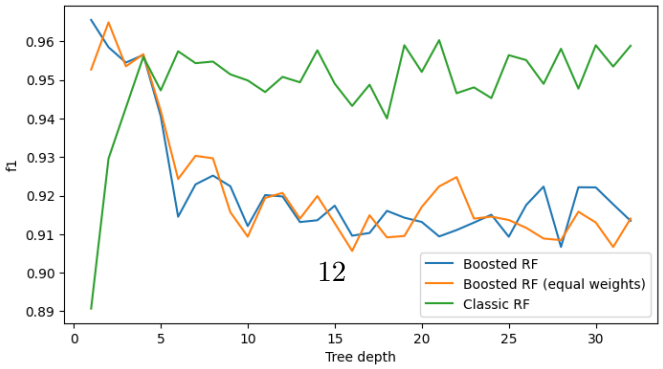
[24]:
```
from matplotlib import pyplot as plt

fig, axes = plt.subplots(5, 1, sharex=True)
fig.set_size_inches(8, 25)
for measure, ax in zip(measures, axes):
    _y = [r[f"test_{measure}"] for r in res]
    _y_ref = [r[f"test_{measure}"] for r in res_ref]
    _y_no_weights = [r[f"test_{measure}"] for r in res_no_weights]
    ax.plot(tree_depths, _y, label="Boosted RF")
    ax.plot(tree_depths, _y_no_weights, label="Boosted RF (equal weights)")
    ax.plot(tree_depths, _y_ref, label="Classic RF")
    ax.set_xlabel("Tree depth")
    ax.set_ylabel(measure)
    ax.legend()

plt.show()
```


accuracy
Tree depth
Boosted RF
Boosted RF (equal weights)
Classic RF


precision
Tree depth
Boosted RF
Boosted RF (equal weights)
Classic RF


recall
Tree depth
Boosted RF
Boosted RF (equal weights)
Classic RF


roc_auc
Tree depth
Boosted RF
Boosted RF (equal weights)
Classic RF


f1
Tree depth
Boosted RF
Boosted RF (equal weights)
Classic RF

Stacking

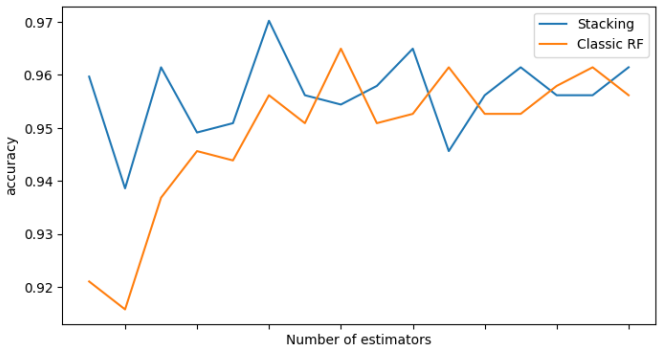
[25]:
```
from sklearn.ensemble import StackingClassifier

clf = StackingClassifier(estimators=[(f"tree{i}", RandomForestClassifier()) for␣
 ↪i in range(16)],
                         final_estimator=RandomForestClassifier())

X = breast_cancer_features_df
breast_cancer_lb_diagnosis = LabelBinarizer()

y = np.squeeze(breast_cancer_lb_diagnosis.
 ↪fit_transform(breast_cancer_classes_df))

evaluate(clf, X, y, scoring=['accuracy', 'precision', 'recall', 'roc_auc',␣
 ↪'f1'])
```

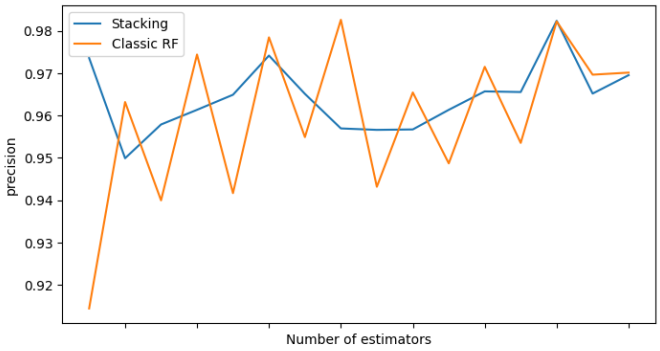
[25]:
```
{'fit_time': 15.19865641593933,
 'score_time': 0.1616374969482422,
 'test_accuracy': 0.956140350877193,
 'train_accuracy': 0.9982417582417582,
 'test_precision': 0.969563887570532,
 'train_precision': 1.0,
 'test_recall': 0.9175271931425899,
 'train_recall': 0.9952438782242208,
 'test_roc_auc': 0.9894386455594674,
 'train_roc_auc': 0.9999958202716822,
 'test_f1': 0.9414258270465167,
 'train_f1': 0.9976113970689516}
```

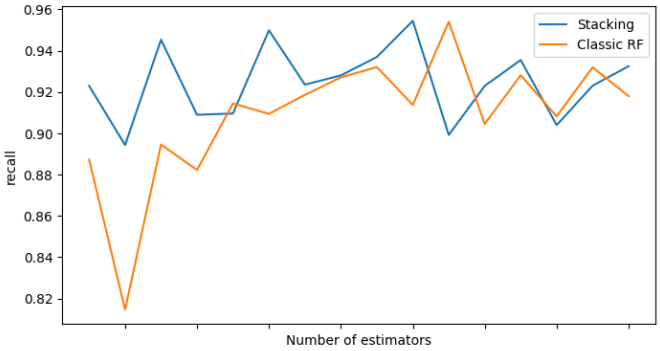
[26]:
```
num_estimators = list(range(1, 17))
measures = ['accuracy', 'precision', 'recall', 'roc_auc', 'f1']
res = [evaluate(StackingClassifier(estimators=[(f"tree{i}",␣
 ↪RandomForestClassifier()) for i in range(n)],
                                    final_estimator=RandomForestClassifier()),␣
 ↪X, y, scoring=measures) for n in
      num_estimators]
```

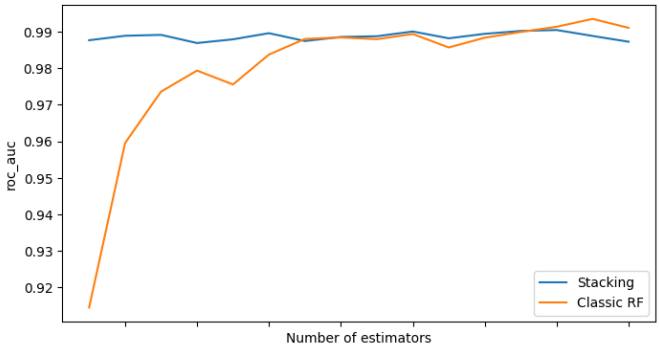
[28]:
```
res_ref = [evaluate(RandomForestClassifier(
    max_depth=None,
    n_estimators=num_estimator,
    n_jobs=8
), X, y, scoring=measures) for num_estimator in num_estimators]
```

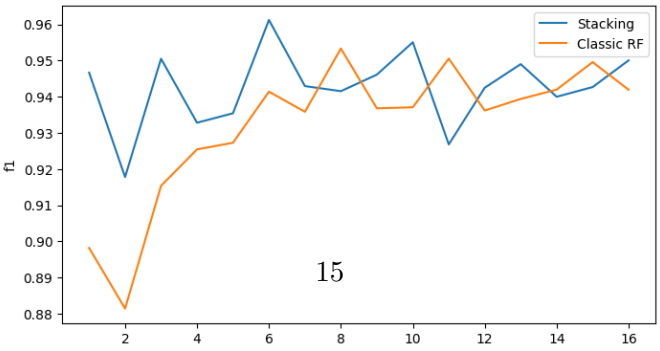
[36]:
```
fig, axes = plt.subplots(5, 1, sharex=True)
fig.set_size_inches(8, 25)
for measure, ax in zip(measures, axes):
    _y = [r[f"test_{measure}"] for r in res]
    _y_ref = [r[f"test_{measure}"] for r in res_ref]
    ax.plot(num_estimators, _y, label="Stacking")
    ax.plot(num_estimators, _y_ref, label="Classic RF")
    ax.set_xlabel("Number of estimators")
    ax.set_ylabel(measure)
    ax.legend()

plt.show()
```


Stacking
Classic RF
accuracy
0.97
0.96
0.95
0.94
0.93
0.92
Number of estimators


Stacking
Classic RF
precision
0.98
0.97
0.96
0.95
0.94
0.93
0.92
Number of estimators


Stacking
Classic RF
recall
0.96
0.94
0.92
0.90
0.88
0.86
0.84
0.82
Number of estimators


Stacking
Classic RF
roc_auc
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0.92
Number of estimators


Stacking
Classic RF
f1
0.96
0.95
0.94
0.93
0.92
0.91
0.90
0.89
0.88
2
4
6
8
10
12
14
16
Number of estimators

[ ]:
```
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression

clf = StackingClassifier(estimators=[
    ("nb", GaussianNB()),
    ("tree", DecisionTreeClassifier()),
    ("rf", RandomForestClassifier()),
    ("knn", KNeighborsClassifier()),
    ("svm", LinearSVC()),
    ("lr", LogisticRegression())
], final_estimator=RandomForestClassifier())

X = breast_cancer_features_df
breast_cancer_lb_diagnosis = LabelBinarizer()

y = np.squeeze(breast_cancer_lb_diagnosis.
 ↪fit_transform(breast_cancer_classes_df))

evaluate(clf, X, y, scoring=['accuracy', 'precision', 'recall', 'roc_auc',␣
 ↪'f1'])
```

[51]:
```
res_single = evaluate(clf, X, y, scoring=['accuracy', 'precision', 'recall',␣
 ↪'roc_auc', 'f1'])
res_variety = [res_single] * len(num_estimators)
```

```
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
```

```
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
```

```
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
```

```
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
```

```
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
```

```
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
```

```
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
```

```
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dual`
will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly
to suppress the warning.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
```

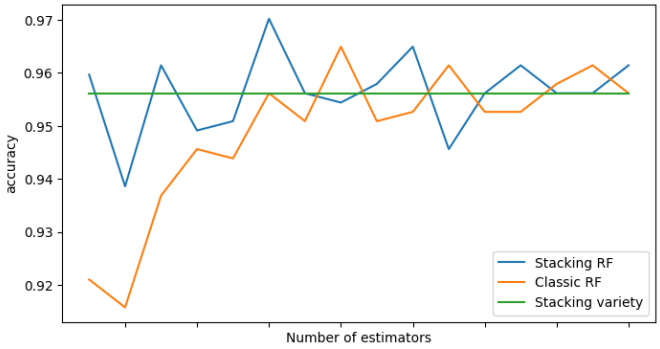
```
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to
converge, increase the number of iterations.
  warnings.warn(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
```

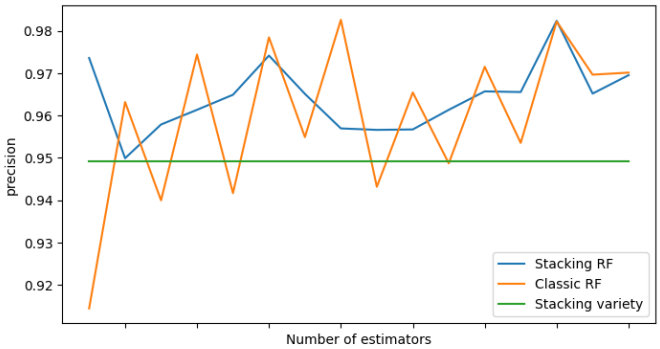
```
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

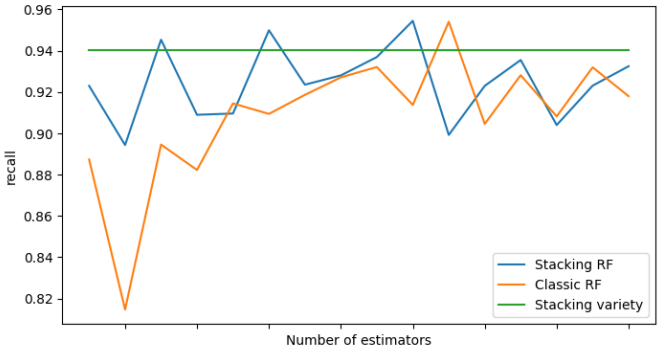
```
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

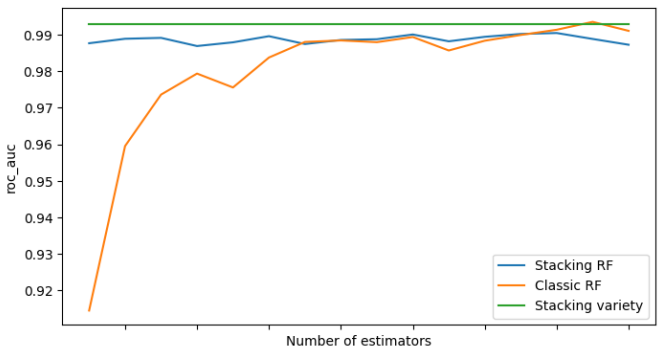
```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

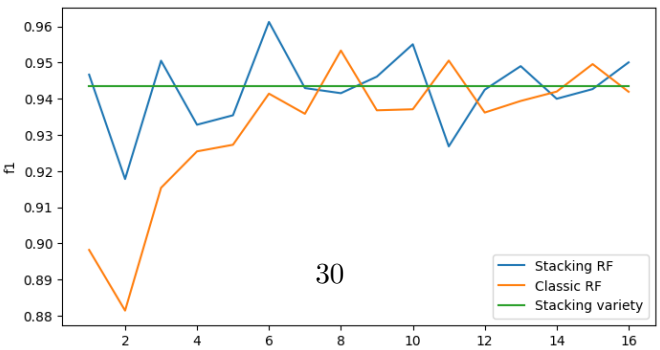
[54]:
```
fig, axes = plt.subplots(5, 1, sharex=True)
fig.set_size_inches(8, 25)
for measure, ax in zip(measures, axes):
    _y_rf = [r[f"test_{measure}"] for r in res]
    _y_variety = [r[f"test_{measure}"] for r in res_variety]
    _y_ref = [r[f"test_{measure}"] for r in res_ref]
    ax.plot(num_estimators, _y_rf, label="Stacking RF")
    ax.plot(num_estimators, _y_ref, label="Classic RF")
    ax.plot(num_estimators, _y_variety, label="Stacking variety")
    ax.set_xlabel("Number of estimators")
    ax.set_ylabel(measure)
    ax.legend()

plt.show()
```


accuracy
Number of estimators
Stacking RF
Classic RF
Stacking variety


precision
Number of estimators
Stacking RF
Classic RF
Stacking variety


recall
Number of estimators
Stacking RF
Classic RF
Stacking variety


roc_auc
Number of estimators
Stacking RF
Classic RF
Stacking variety


f1
Number of estimators
Stacking RF
Classic RF
Stacking variety

1.1 Iterative RF

[104]:
```
from sklearn.metrics import confusion_matrix
from sklearn.base import BaseEstimator

class IterativeRandomForrest(BaseEstimator):
    def __init__(
            self,
            *,
            iter_count=2,
            n_estimators=2,
            max_depth=None,
    ):

        self.iter_count = 2
        self.n_estimators = n_estimators
        self.max_depth = max_depth
        self._fitted_estimator = None

    def _get_estimator(self, weights):
        return RandomForestClassifier(
            n_estimators=self.n_estimators,
            max_depth=self.max_depth,
            n_jobs=8,
            class_weight=weights
        )

    def _get_accuracy_per_class(self, y, y_pred):
        c_matrix = confusion_matrix(y, y_pred)
        return c_matrix.diagonal() / c_matrix.sum(axis=1)

    def _get_weights(self, class_accuracies, y):
        unique_classes = np.unique(y)
        weights = {}
        for class_idx, class_accuracy in enumerate(class_accuracies):
            weights[unique_classes[class_idx]] = 1 - class_accuracy
        return weights

    def _get_initial_weights(self, y):
        unique_classes = np.unique(y)
        weights = {}
        for class_name in unique_classes:
            weights[class_name] = 1
        return weights
```

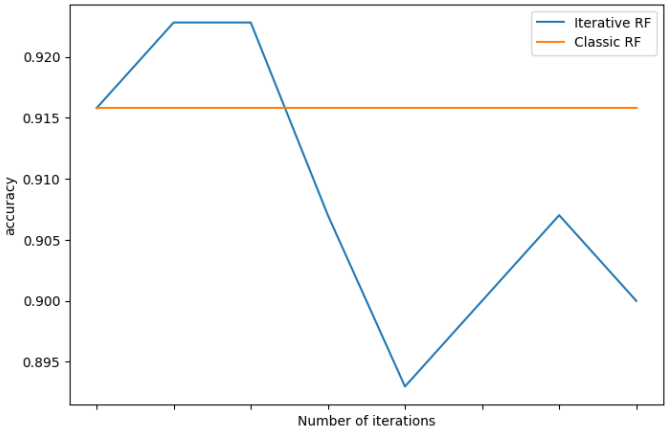
```
    def fit(self, X, y):
        weights = self._get_initial_weights(y)

        for iteration_idx in range(self.iter_count):
            estimator = self._get_estimator(weights)
            estimator.fit(X, y)
            y_pred = estimator.predict(X)
            class_accuracies = self._get_accuracy_per_class(y, y_pred)
            weights = self._get_weights(class_accuracies, y)
        self._fitted_estimator = estimator

    def predict(self, X):
        return self._fitted_estimator.predict(X)

    def decision_function(self, X):
        # called for roc_auc_score
        raise NotImplementedError("This IterativeRandomForrest instance is not␣
↪fitted yet")


    def predict_proba(self, X):
        # called for roc_auc_score
        raise NotImplementedError("This IterativeRandomForrest instance is not␣
↪fitted yet")

clf = IterativeRandomForrest(iter_count=2, n_estimators=2, max_depth=2)
clf.fit(X, y)
clf.predict(X)
clf.get_params()
evaluate(clf, X, y, scoring=['accuracy', 'precision', 'recall', 'f1'])
```

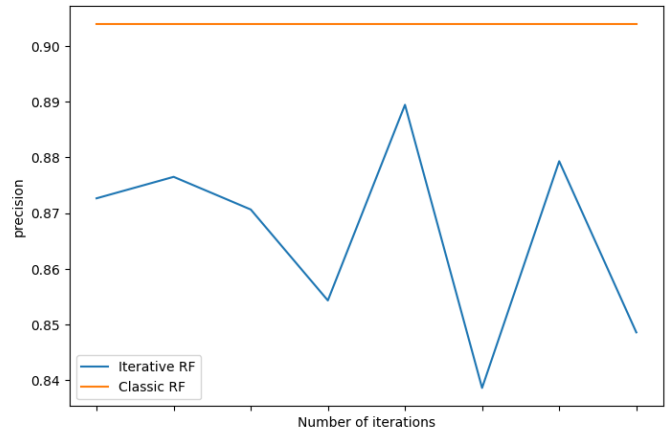
[104]:
```
{'fit_time': 0.06289176940917969,
 'score_time': 0.01744365692138672,
 'test_accuracy': 0.9087719298245615,
 'train_accuracy': 0.9261538461538462,
 'test_precision': 0.8598644705347851,
 'train_precision': 0.8698541768515575,
 'test_recall': 0.9231330930751069,
 'train_recall': 0.9460505956384206,
 'test_f1': 0.8882752413680249,
 'train_f1': 0.9043734538199694}
```

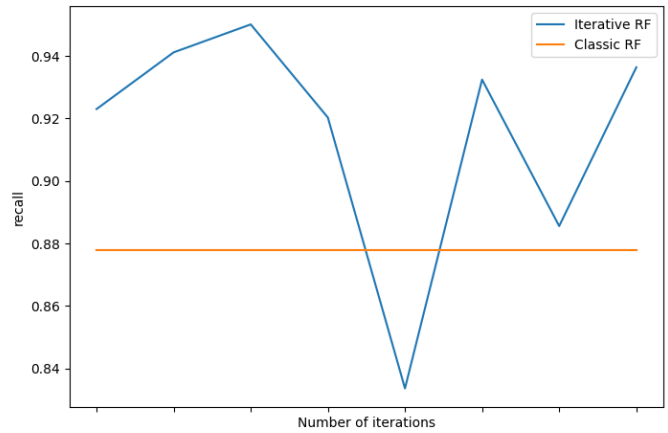
[105]:
```
NUM_ESTIMATORS = 2
MAX_DEPTH = 2
iteration_counts = list(range(1, 9))
measures = ['accuracy', 'precision', 'recall', 'f1']
```

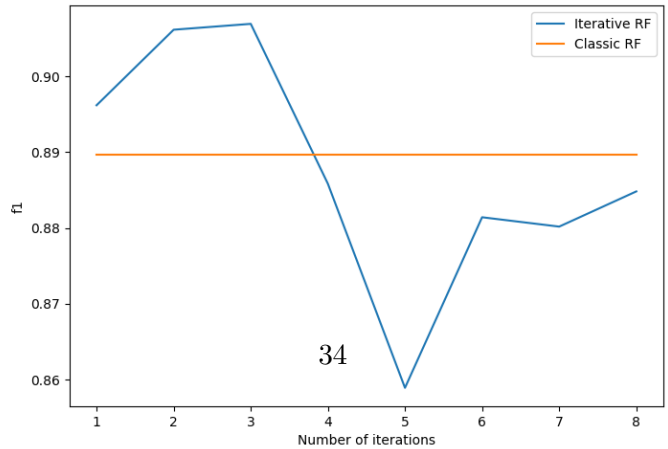
```
res = [evaluate(IterativeRandomForrest(n_estimators=NUM_ESTIMATORS,␣
 ↪max_depth=MAX_DEPTH, iter_count=iter_count), X, y, scoring=measures) for␣
 ↪iter_count in iteration_counts]

res_ref_single = evaluate(RandomForestClassifier(
    max_depth=MAX_DEPTH,
    n_estimators=NUM_ESTIMATORS,
    n_jobs=8
), X, y, scoring=measures)
res_ref = [res_ref_single] * len(iteration_counts)
```

[106]:
```
fig, axes = plt.subplots(4, 1, sharex=True)
fig.set_size_inches(8, 25)
for measure, ax in zip(measures, axes):
    _y_iter = [r[f"test_{measure}"] for r in res]
    _y_ref = [r[f"test_{measure}"] for r in res_ref]
    ax.plot(iteration_counts, _y_iter, label="Iterative RF")
    ax.plot(iteration_counts, _y_ref, label="Classic RF")
    ax.set_xlabel("Number of iterations")
    ax.set_ylabel(measure)
    ax.legend()

plt.show()
```


Iterative RF
Classic RF
accuracy
0.920
0.915
0.910
0.905
0.900
0.895
Number of iterations


precision
0.90
0.89
0.88
0.87
0.86
0.85
0.84
Iterative RF
Classic RF
Number of iterations


Iterative RF
Classic RF
recall
0.94
0.92
0.90
0.88
0.86
0.84
Number of iterations


Iterative RF
Classic RF
f1
0.90
0.89
0.88
0.87
0.86
1
2
3
4
5
6
7
8
Number of iterations

[ ]:

[107]:
```
from ucimlrepo import fetch_ucirepo

# fetch dataset
statlog_german_credit_data = fetch_ucirepo(id=144)

# data (as pandas dataframes)
X = statlog_german_credit_data.data.features
y = statlog_german_credit_data.data.targets
```

```
{'uci_id': 144, 'name': 'Statlog (German Credit Data)', 'repository_url':
'https://archive.ics.uci.edu/dataset/144/statlog+german+credit+data',
'data_url': 'https://archive.ics.uci.edu/static/public/144/data.csv',
'abstract': 'This dataset classifies people described by a set of attributes as
good or bad credit risks. Comes in two formats (one all numeric). Also comes
with a cost matrix', 'area': 'Social Science', 'tasks': ['Classification'],
'characteristics': ['Multivariate'], 'num_instances': 1000, 'num_features': 20,
'feature_types': ['Categorical', 'Integer'], 'demographics': ['Other', 'Marital
Status', 'Age', 'Occupation'], 'target_col': ['class'], 'index_col': None,
'has_missing_values': 'no', 'missing_values_symbol': None,
'year_of_dataset_creation': 1994, 'last_updated': 'Thu Aug 10 2023',
'dataset_doi': '10.24432/C5NC77', 'creators': ['Hans Hofmann'], 'intro_paper':
None, 'additional_info': {'summary': 'Two datasets are provided.  the original
dataset, in the form provided by Prof. Hofmann, contains categorical/symbolic
attributes and is in the file "german.data".   \r\n \r\nFor algorithms that need
numerical attributes, Strathclyde University produced the file "german.data-
numeric".  This file has been edited and several indicator variables added to
make it suitable for algorithms which cannot cope with categorical variables.
Several attributes that are ordered categorical (such as attribute 17) have been
coded as integer.    This was the form used by StatLog.\r\n\r\nThis dataset
requires use of a cost matrix (see below)\r\n\r\n … 1
2\r\n----------------------------\r\n  1   0
1\r\n-----------------------\r\n  2   5        0\r\n\r\n(1 = Good,  2 =
Bad)\r\n\r\nThe rows represent the actual classification and the columns the
predicted classification.\r\n\r\nIt is worse to class a customer as good when
they are bad (5), than it is to class a customer as bad when they are good
(1).\r\n', 'purpose': None, 'funded_by': None, 'instances_represent': None,
'recommended_data_splits': None, 'sensitive_data': None,
'preprocessing_description': None, 'variable_info': 'Attribute 1:  (qualitative)
\r\n Status of existing checking account\r\n              A11 :      … <    0
DM\r\n\t       A12 : 0 <= … <  200 DM\r\n\t       A13 :      … >= 200 DM /
salary assignments for at least 1 year\r\n               A14 : no checking
account\r\n\r\nAttribute 2:  (numerical)\r\n\t      Duration in
month\r\n\r\nAttribute 3:  (qualitative)\r\n\t      Credit history\r\n\t
```

A30 : no credits taken/ all credits paid back duly\r\n A31 : all credits at this bank paid back duly\r\n\t A32 : existing credits paid back duly till now\r\n A33 : delay in paying off in the past\r\n\t A34 : critical account/ other credits existing (not at this bank)\r\n\r\nAttribute 4: (qualitative)\r\n\t Purpose\r\n\t A40 : car (new)\r\n\t A41 : car (used)\r\n\t A42 : furniture/equipment\r\n\t A43 : radio/television\r\n\t A44 : domestic appliances\r\n\t A45 : repairs\r\n\t A46 : education\r\n\t A47 : (vacation - does not exist?)\r\n\t A48 : retraining\r\n\t A49 : business\r\n\t A410 : others\r\n\r\nAttribute 5: (numerical)\r\n\t Credit amount\r\n\r\nAttibute 6: (qualitative)\r\n\t Savings account/bonds\r\n\t A61 : … < 100 DM\r\n\t A62 : 100 <= … < 500 DM\r\n\t A63 : 500 <= … < 1000 DM\r\n\t A64 : .. >= 1000 DM\r\n A65 : unknown/ no savings account\r\n\r\nAttribute 7: (qualitative)\r\n\t Present employment since\r\n\t A71 : unemployed\r\n\t A72 : … < 1 year\r\n\t A73 : 1 <= … < 4 years \r\n\t A74 : 4 <= … < 7 years\r\n\t A75 : .. >= 7 years\r\n\r\nAttribute 8: (numerical)\r\n\t Installment rate in percentage of disposable income\r\n\r\nAttribute 9: (qualitative)\r\n\t Personal status and sex\r\n\t A91 : male : divorced/separated\r\n\t A92 : female : divorced/separated/married\r\n A93 : male : single\r\n\t A94 : male : married/widowed\r\n\t A95 : female : single\r\n\r\nAttribute 10: (qualitative)\r\n\t Other debtors / guarantors\r\n\t A101 : none\r\n\t A102 : co-applicant\r\n\t A103 : guarantor\r\n\r\nAttribute 11: (numerical)\r\n\t Present residence since\r\n\r\nAttribute 12: (qualitative)\r\n\t Property\r\n\t A121 : real estate\r\n\t A122 : if not A121 : building society savings agreement/ life insurance\r\n A123 : if not A121/A122 : car or other, not in attribute 6\r\n\t A124 : unknown / no property\r\n\r\nAttribute 13: (numerical)\r\n\t Age in years\r\n\r\nAttribute 14: (qualitative)\r\n\t Other installment plans \r\n\t A141 : bank\r\n\t A142 : stores\r\n\t A143 : none\r\n\r\nAttribute 15: (qualitative)\r\n\t Housing\r\n\t A151 : rent\r\n\t A152 : own\r\n\t A153 : for free\r\n\r\nAttribute 16: (numerical)\r\n Number of existing credits at this bank\r\n\r\nAttribute 17: (qualitative)\r\n\t Job\r\n\t A171 : unemployed/ unskilled - non-resident\r\n\t A172 : unskilled - resident\r\n\t A173 : skilled employee / official\r\n\t A174 : management/ self-employed/\r\n\t\t highly qualified employee/ officer\r\n\r\nAttribute 18: (numerical)\r\n\t Number of people being liable to provide maintenance for\r\n\r\nAttribute 19: (qualitative)\r\n\t Telephone\r\n\t A191 : none\r\n\t A192 : yes, registered under the customers name\r\n\r\nAttribute 20: (qualitative)\r\n\t foreign worker\r\n\t A201 : yes\r\n\t A202 : no\r\n', 'citation': None}}

	name	role	type	demographic	\
0	Attribute1	Feature	Categorical	None	
1	Attribute2	Feature	Integer	None	
2	Attribute3	Feature	Categorical	None	
3	Attribute4	Feature	Categorical	None	

```
4    Attribute5  Feature      Integer            None
5    Attribute6  Feature  Categorical            None
6    Attribute7  Feature  Categorical           Other
7    Attribute8  Feature      Integer            None
8    Attribute9  Feature  Categorical  Marital Status
9   Attribute10  Feature  Categorical            None
10  Attribute11  Feature      Integer            None
11  Attribute12  Feature  Categorical            None
12  Attribute13  Feature      Integer             Age
13  Attribute14  Feature  Categorical            None
14  Attribute15  Feature  Categorical           Other
15  Attribute16  Feature      Integer            None
16  Attribute17  Feature  Categorical      Occupation
17  Attribute18  Feature      Integer            None
18  Attribute19  Feature       Binary            None
19  Attribute20  Feature       Binary           Other
20        class   Target       Binary            None

                                          description   units missing_values
0                Status of existing checking account    None             no
1                                            Duration  months             no
2                                      Credit history    None             no
3                                             Purpose    None             no
4                                       Credit amount    None             no
5                                 Savings account/bonds    None             no
6                               Present employment since    None             no
7   Installment rate in percentage of disposable i…    None             no
8                               Personal status and sex    None             no
9                            Other debtors / guarantors    None             no
10                              Present residence since    None             no
11                                            Property    None             no
12                                                 Age   years             no
13                              Other installment plans    None             no
14                                            Housing    None             no
15           Number of existing credits at this bank    None             no
16                                                 Job    None             no
17  Number of people being liable to provide maint…    None             no
18                                           Telephone    None             no
19                                      foreign worker    None             no
20                                1 = Good, 2 = Bad    None             no
```

[108]:
```
from sklearn.preprocessing import LabelEncoder, LabelBinarizer

statlog_german_credit_data_df = pd.DataFrame.
  ↪from_dict(statlog_german_credit_data.data.features)
statlog_german_credit_data_df['class'] = statlog_german_credit_data.data.
  ↪targets['class']
```

```
statlog_german_credit_data_df['class'] = statlog_german_credit_data_df['class'].
 ↪astype(str)

encoded_features = statlog_german_credit_data_df.columns
label_encoders = {feature: LabelEncoder() for feature in encoded_features}
for feature, encoder in label_encoders.items():
    statlog_german_credit_data_df[feature] = encoder.
 ↪fit_transform(statlog_german_credit_data_df[feature])

statlog_german_credit_data_df
```

[108]:

	Attribute1	Attribute2	Attribute3	Attribute4	Attribute5	Attribute6	\
0	0	2	4	4	142	4	
1	1	29	2	4	770	0	
2	3	8	4	7	390	0	
3	0	26	2	3	848	0	
4	0	17	3	0	734	0	
..	…	…	…	…	…	…	
995	3	8	2	3	310	0	
996	0	21	2	1	661	0	
997	3	8	2	4	69	0	
998	0	27	2	4	332	0	
999	1	27	4	1	711	1	

	Attribute7	Attribute8	Attribute9	Attribute10	…	Attribute12	\
0	4	3	2	0	…	0	
1	2	1	1	0	…	0	
2	3	1	2	0	…	0	
3	3	1	2	2	…	1	
4	2	2	2	0	…	3	
..	…	…	…	…	…	…	
995	3	2	1	0	…	0	
996	2	3	0	0	…	1	
997	4	3	2	0	…	2	
998	2	3	2	0	…	3	
999	0	2	2	0	…	2	

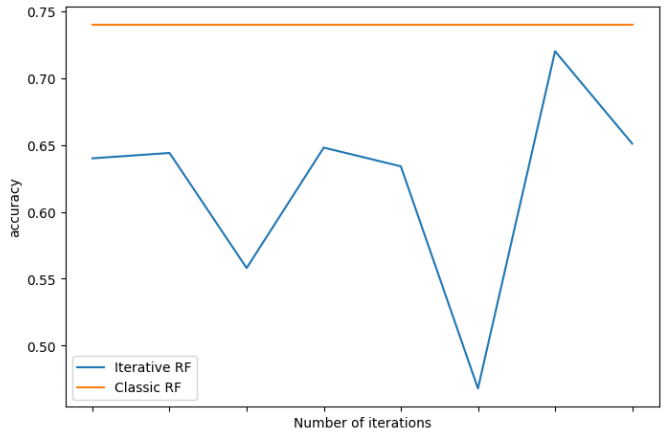
	Attribute13	Attribute14	Attribute15	Attribute16	Attribute17	\
0	48	2	1	1	2	
1	3	2	1	0	2	
2	30	2	1	0	1	
3	26	2	2	0	2	
4	34	2	2	1	2	
..	…	…	…	…	…	
995	12	2	1	0	1	
996	21	2	1	0	3	
997	19	2	1	0	2	

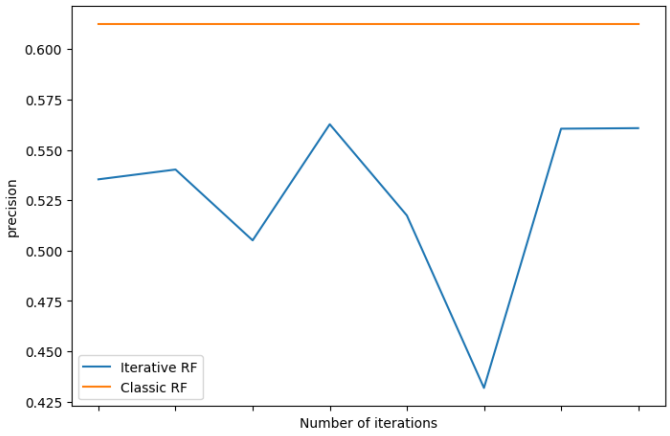
```
998            4            2            2            0            2
999            8            2            1            0            2

     Attribute18  Attribute19  Attribute20  class
0              0            1            0      0
1              0            0            0      1
2              1            0            0      0
3              1            0            0      0
4              1            0            0      1
..           …            …          …    …
995            0            0            0      0
996            0            1            0      0
997            0            0            0      0
998            0            1            0      1
999            0            0            0      0

[1000 rows x 21 columns]
```

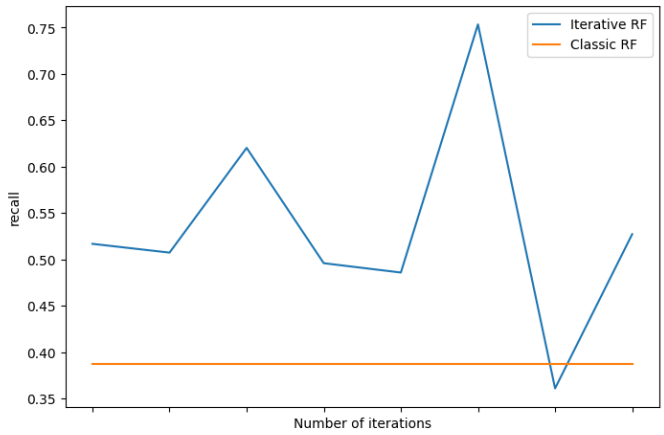
[110]:
```
X = statlog_german_credit_data_df.drop(['class'], axis='columns')
y = statlog_german_credit_data_df['class']
```

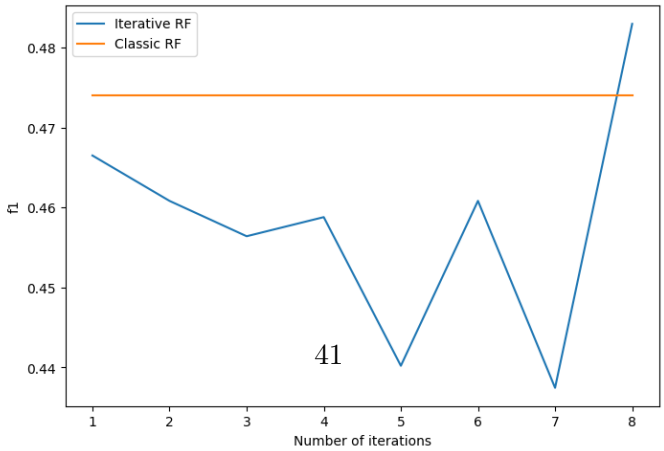
[120]:
```
NUM_ESTIMATORS = 10
MAX_DEPTH = 15
iteration_counts = list(range(1, 9))
measures = ['accuracy', 'precision', 'recall', 'f1']
res = [evaluate(IterativeRandomForrest(n_estimators=NUM_ESTIMATORS,␣
 ↪max_depth=MAX_DEPTH, iter_count=iter_count), X, y, scoring=measures) for␣
 ↪iter_count in iteration_counts]

res_ref_single = evaluate(RandomForestClassifier(
    max_depth=MAX_DEPTH,
    n_estimators=NUM_ESTIMATORS,
    n_jobs=8
), X, y, scoring=measures)
res_ref = [res_ref_single] * len(iteration_counts)




fig, axes = plt.subplots(4, 1, sharex=True)
fig.set_size_inches(8, 25)
for measure, ax in zip(measures, axes):
    _y_iter = [r[f"test_{measure}"] for r in res]
    _y_ref = [r[f"test_{measure}"] for r in res_ref]
    ax.plot(iteration_counts, _y_iter, label="Iterative RF")
    ax.plot(iteration_counts, _y_ref, label="Classic RF")
    ax.set_xlabel("Number of iterations")
```

```
    ax.set_ylabel(measure)
    ax.legend()

plt.show()
```


accuracy
Number of iterations
Iterative RF
Classic RF


precision
Number of iterations
Iterative RF
Classic RF


recall
Number of iterations
Iterative RF
Classic RF


f1
Number of iterations
Iterative RF
Classic RF

[171]:
```
class IterativeRandomForrestSampleBased(BaseEstimator):
    def __init__(
            self,
            *,
            iter_count=2,
            n_estimators=2,
            max_depth=None,
    ):

        self.iter_count = iter_count
        self.n_estimators = n_estimators
        self.max_depth = max_depth
        self._fitted_estimator = None

    def _get_estimator(self):
        return RandomForestClassifier(
            n_estimators=self.n_estimators,
            max_depth=self.max_depth,
            n_jobs=8
        )


    def fit(self, X, y):
        for iteration_idx in range(self.iter_count):
            estimator = self._get_estimator()
            estimator.fit(X, y)
            y_pred = estimator.predict(X)
            misclassified_samples = np.where(y_pred != y)[0]
            X = pd.concat([X, X.iloc[misclassified_samples]])
            y = np.concatenate([y, y[misclassified_samples]])

        self._fitted_estimator = estimator

    def get_fiited_estimator(self):
        return self._fitted_estimator

    def predict(self, X):
        return self._fitted_estimator.predict(X)

    def decision_function(self, X):
        # called for roc_auc_score
        raise NotImplementedError("This IterativeRandomForrest instance is not␣
↪fitted yet")
```

```
    def predict_proba(self, X):
        # called for roc_auc_score
        raise NotImplementedError("This IterativeRandomForrest instance is not␣
 ↪fitted yet")

clf = IterativeRandomForrestSampleBased(iter_count=2, n_estimators=10,␣
 ↪max_depth=15)
clf.fit(X, y)
fancy_rf = clf.get_fiited_estimator()
evaluate(fancy_rf, X, y, scoring=['accuracy', 'precision', 'recall', 'f1'])
# evaluate(clf, X, y, scoring=['accuracy', 'precision', 'recall', 'f1'])
```

[171]:
```
{'fit_time': 0.03116297721862793,
 'score_time': 0.021782112121582032,
 'test_accuracy': 0.734,
 'train_accuracy': 0.9807499999999999,
 'test_precision': 0.6241318896157606,
 'train_precision': 0.9965100519539192,
 'test_recall': 0.33318823460535785,
 'train_recall': 0.9391317615112357,
 'test_f1': 0.4290354054880171,
 'train_f1': 0.96685846006811}
```

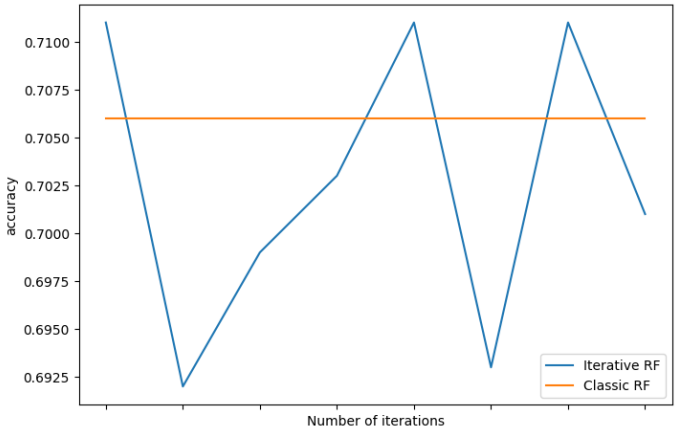
[ ]:

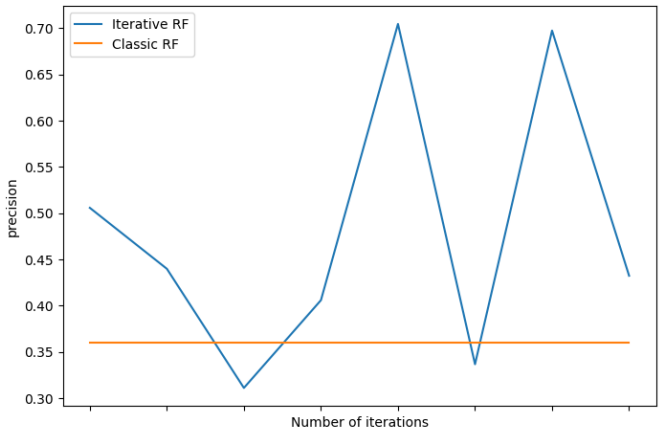
[173]:
```
NUM_ESTIMATORS = 2
MAX_DEPTH = 2
iteration_counts = list(range(1, 9))
measures = ['accuracy', 'precision', 'recall', 'f1']

itertative_rfs = []
for iter_count in iteration_counts:
    clf = IterativeRandomForrestSampleBased(n_estimators=NUM_ESTIMATORS,␣
 ↪max_depth=MAX_DEPTH, iter_count=iter_count)
    clf.fit(X, y)
    itertative_rfs.append(clf.get_fiited_estimator())

res = [evaluate(itertative_rfs[idx], X, y, scoring=measures) for idx,␣
 ↪iter_count in enumerate(iteration_counts)]

res_ref_single = evaluate(RandomForestClassifier(
    max_depth=MAX_DEPTH,
    n_estimators=NUM_ESTIMATORS,
    n_jobs=8
), X, y, scoring=measures)
res_ref = [res_ref_single] * len(iteration_counts)
```

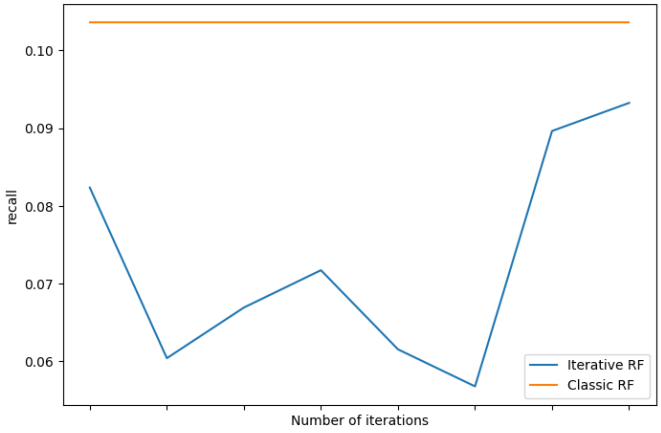
```python
fig, axes = plt.subplots(4, 1, sharex=True)
fig.set_size_inches(8, 25)
for measure, ax in zip(measures, axes):
    _y_iter = [r[f"test_{measure}"] for r in res]
    _y_ref = [r[f"test_{measure}"] for r in res_ref]
    ax.plot(iteration_counts, _y_iter, label="Iterative RF")
    ax.plot(iteration_counts, _y_ref, label="Classic RF")
    ax.set_xlabel("Number of iterations")
    ax.set_ylabel(measure)
    ax.legend()

plt.show()
```

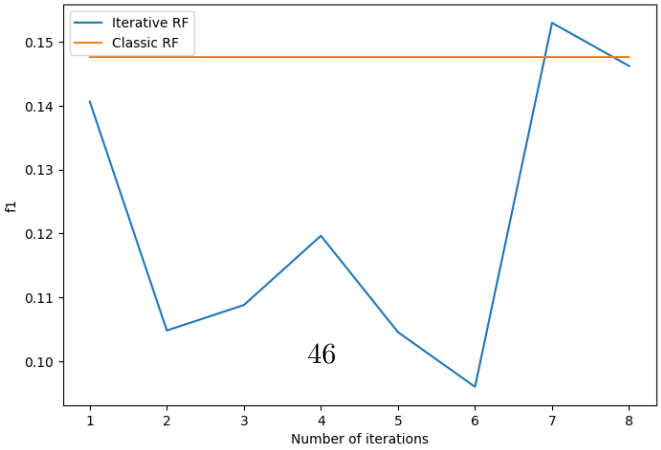
```
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1497: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1497: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1497: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1497: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1497: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1497: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1497: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use
```

```
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1497: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1497: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1497: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1497: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/media/data/coding/uma/.venv/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1497: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```


accuracy
0.7100
0.7075
0.7050
0.7025
0.7000
0.6975
0.6950
0.6925
Iterative RF
Classic RF
Number of iterations


precision
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
Iterative RF
Classic RF
Number of iterations


recall
0.10
0.09
0.08
0.07
0.06
Iterative RF
Classic RF
Number of iterations


f1
0.15
0.14
0.13
0.12
0.11
0.10
Iterative RF
Classic RF
1
2
3
4
5
6
7
8
Number of iterations