Numerical Methods and Machine Learning for Image Processing

Week 4, Class 2: Classification and Regression

October 19, 2021

Damon M. Chandler and Yi Zhang

Last time: Classification and Regression, Part 1a

1. Basic definitions
2. Bayesian classification
    1. Problem formulation
    2. Bayes classification in Excel
    3. Bayes classification in Python

Today: Classification and Regression, Part 1b

1. Bayesian classification of another dataset
2. Decision trees (classification)
3. Random forests (classification)

Today: Classification and Regression, Part 1b

1. Bayesian classification of another dataset
2. Decision trees (classification)
3. Random forests (classification)

https://archive.ics.uci.edu/ml/datasets/banknote+authentication

banknote authentication Data Set

Download: Data Folder, Data Set Description

Abstract: Data were extracted from images that were taken for the evaluation of an authentication procedure for banknotes.

Data Set Characteristics:	Multivariate	**Number of Instances:**	1372	**Area:**	Computer
Attribute Characteristics:	Real	**Number of Attributes:**	5	**Date Donated**	2013-04-16
Associated Tasks:	Classification	**Missing Values?**	N/A	**Number of Web Hits:**	316920

Source:

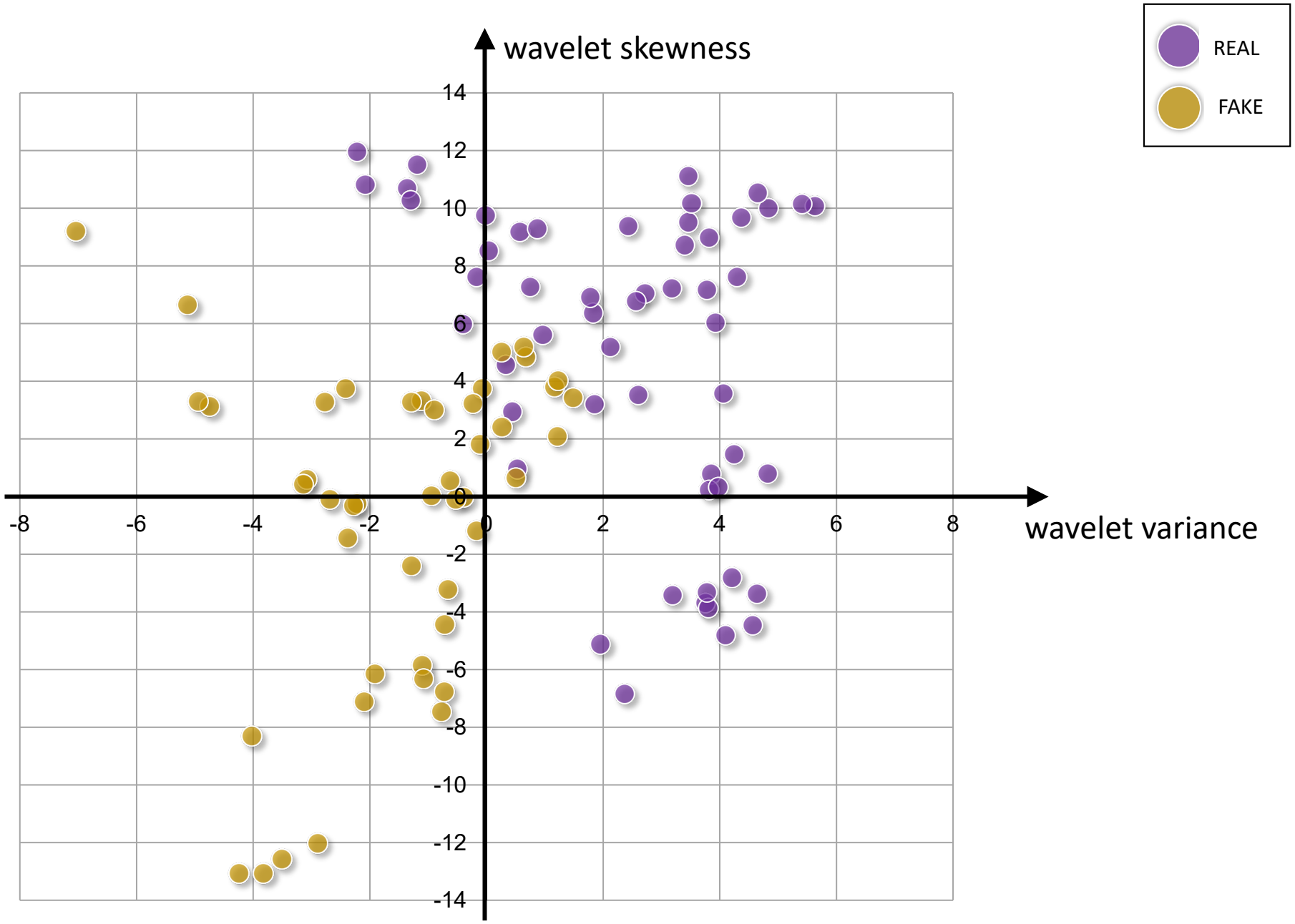
Owner of database: Volker Lohweg (University of Applied Sciences, Ostwestfalen-Lippe, volker.lohweg '@' hs-owl.de)
Donor of database: Helene DÃ¶rksen (University of Applied Sciences, Ostwestfalen-Lippe, helene.doerksen '@' hs-owl.de)
Date received: August, 2012

Data Set Information:

Data were extracted from images that were taken from genuine and forged banknote-like specimens. For digitization, an industrial camera usually used for print inspection was used. The final images have 400x 400 pixels. Due to the object lens and distance to the investigated object gray-scale pictures with a resolution of about 660 dpi were gained. Wavelet Transform tool were used to extract features from images.

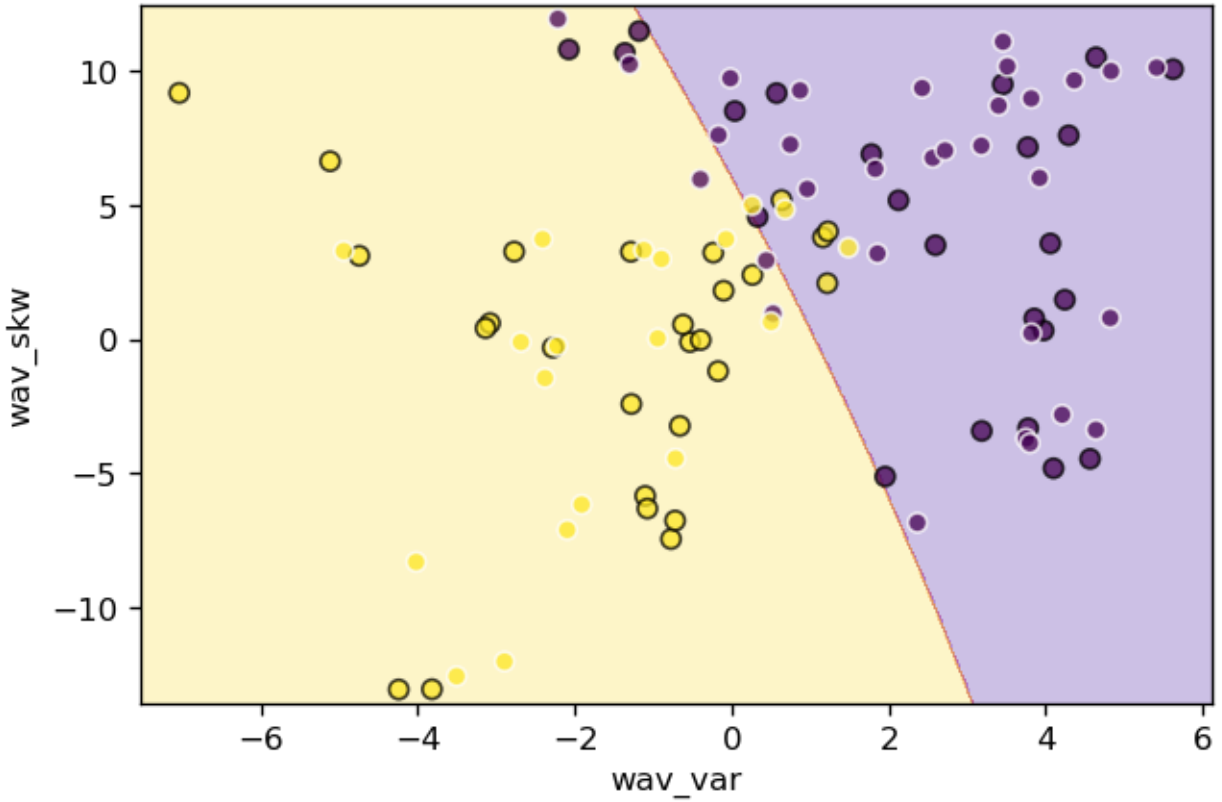
Attribute Information:

1. variance of Wavelet Transformed image (continuous)
2. skewness of Wavelet Transformed image (continuous)
3. curtosis of Wavelet Transformed image (continuous)
4. entropy of image (continuous)
5. class (integer)


wavelet skewness
wavelet variance
REAL
FAKE
14
12
10
8
6
4
2
0
-2
-4
-6
-8
-10
-12
-14
-8
-6
-4
-2
0
2
4
6
8

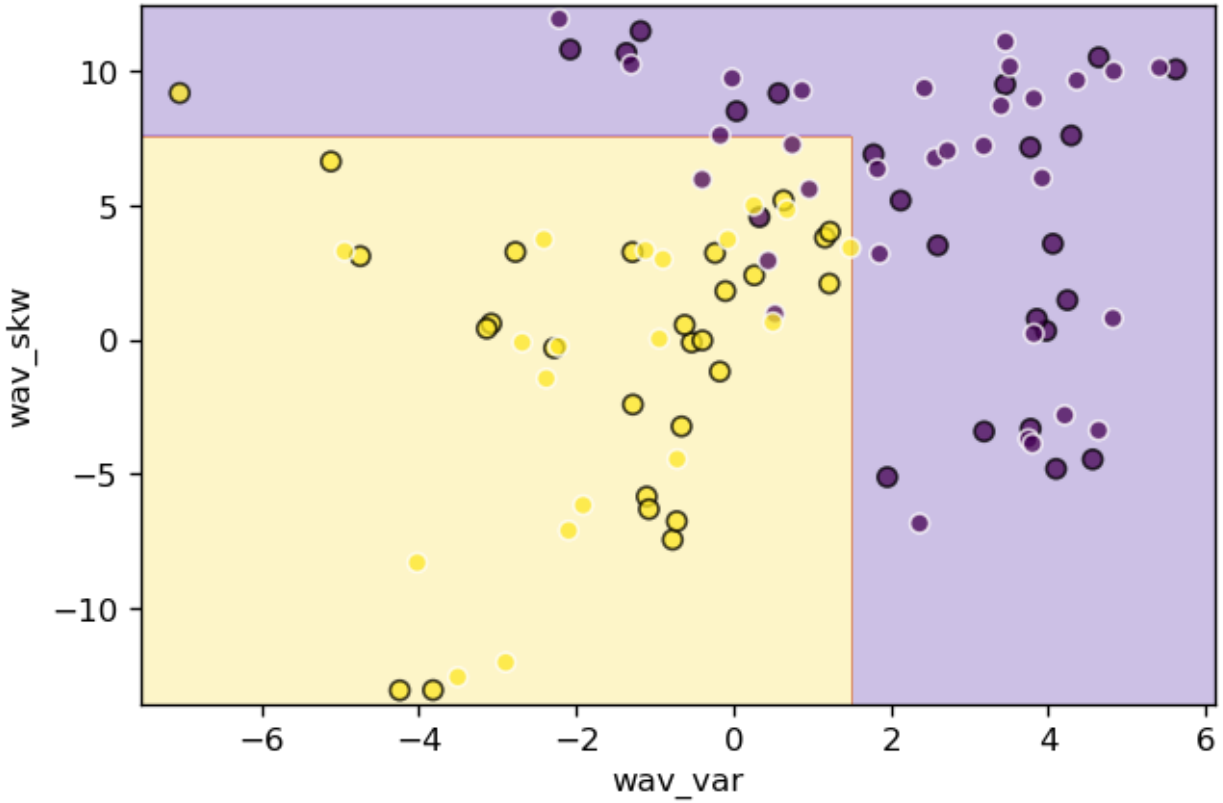
Classification on banknote dataset

Bayes classifier




```
Training accuracy: 0.86
 Testing accuracy: 0.84
```

Decision tree classifier

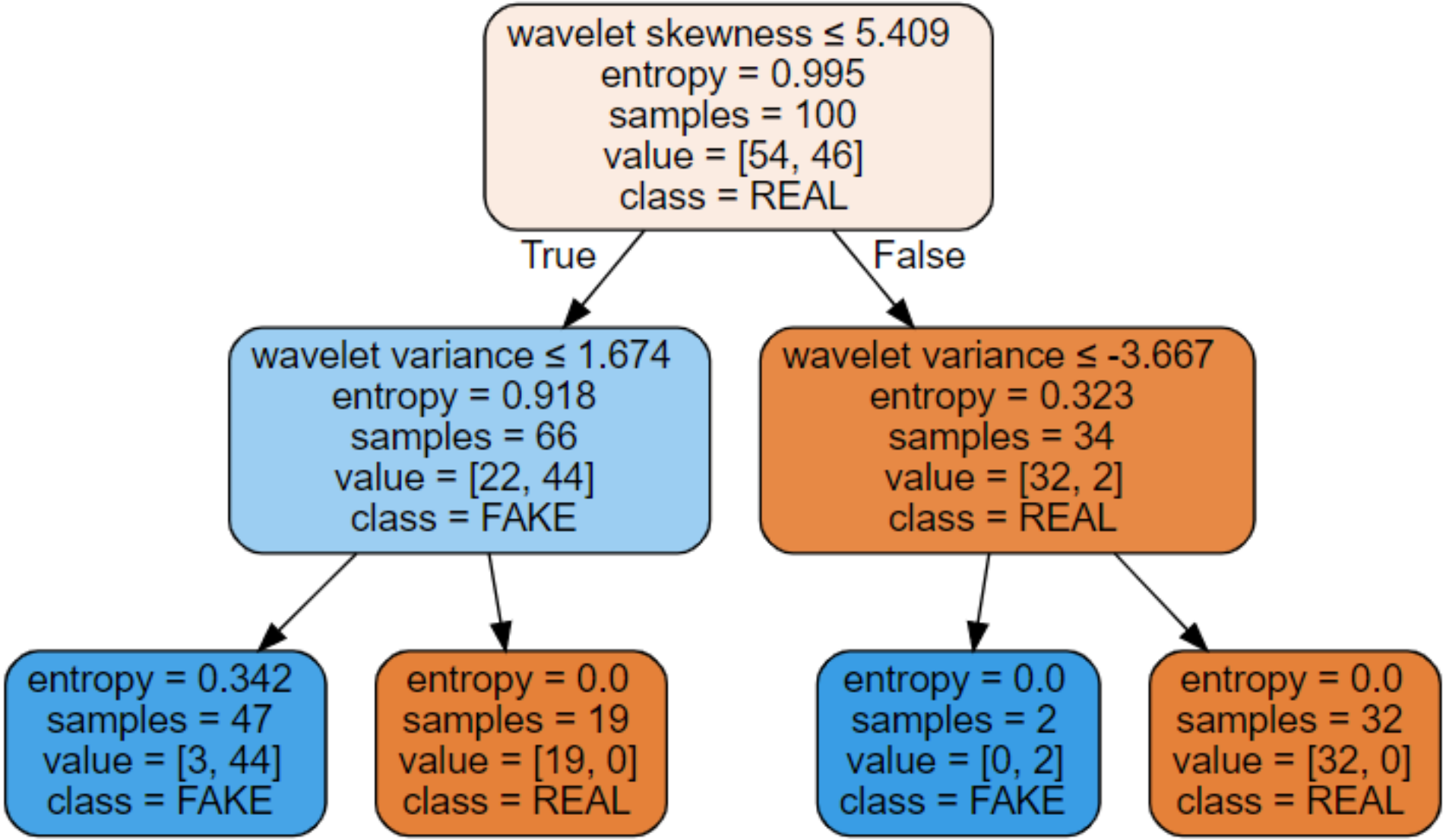



```
Training accuracy: 0.96
 Testing accuracy: 0.90
```

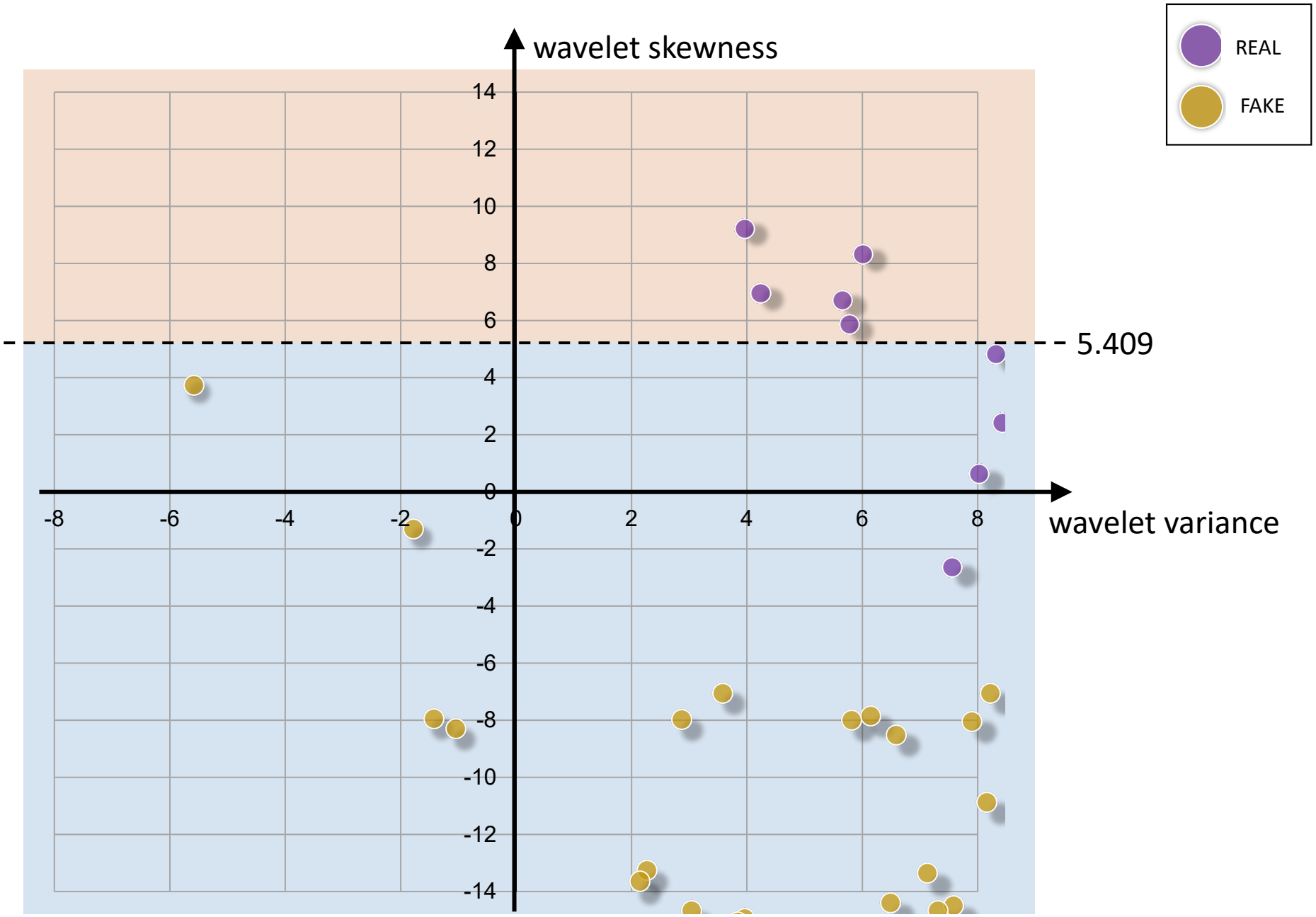
Today: Classification and Regression, Part 1b

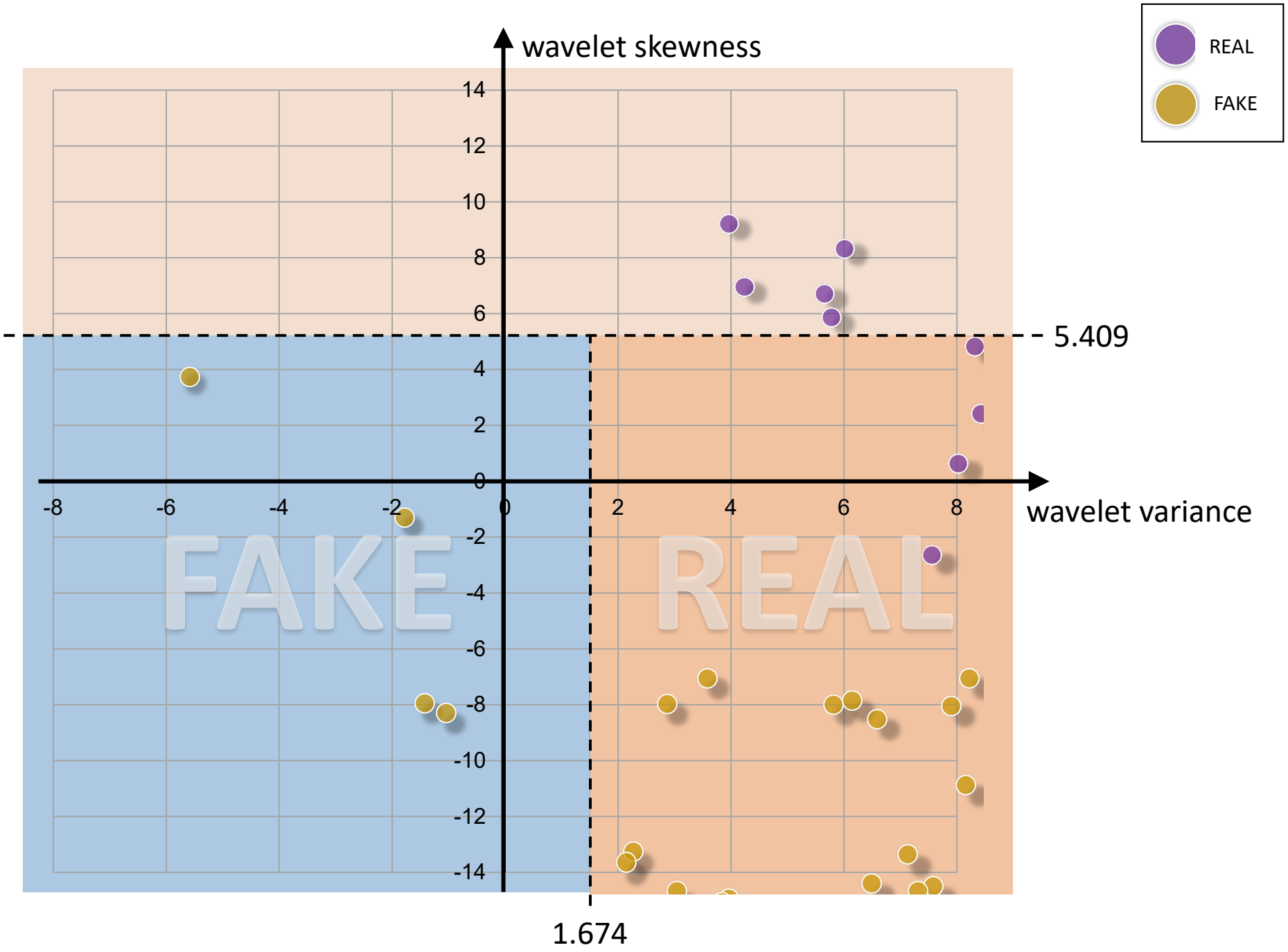
1. Bayesian classification of another dataset
2. Decision trees (classification)
3. Random forests (classification)
4. Other simple classifiers

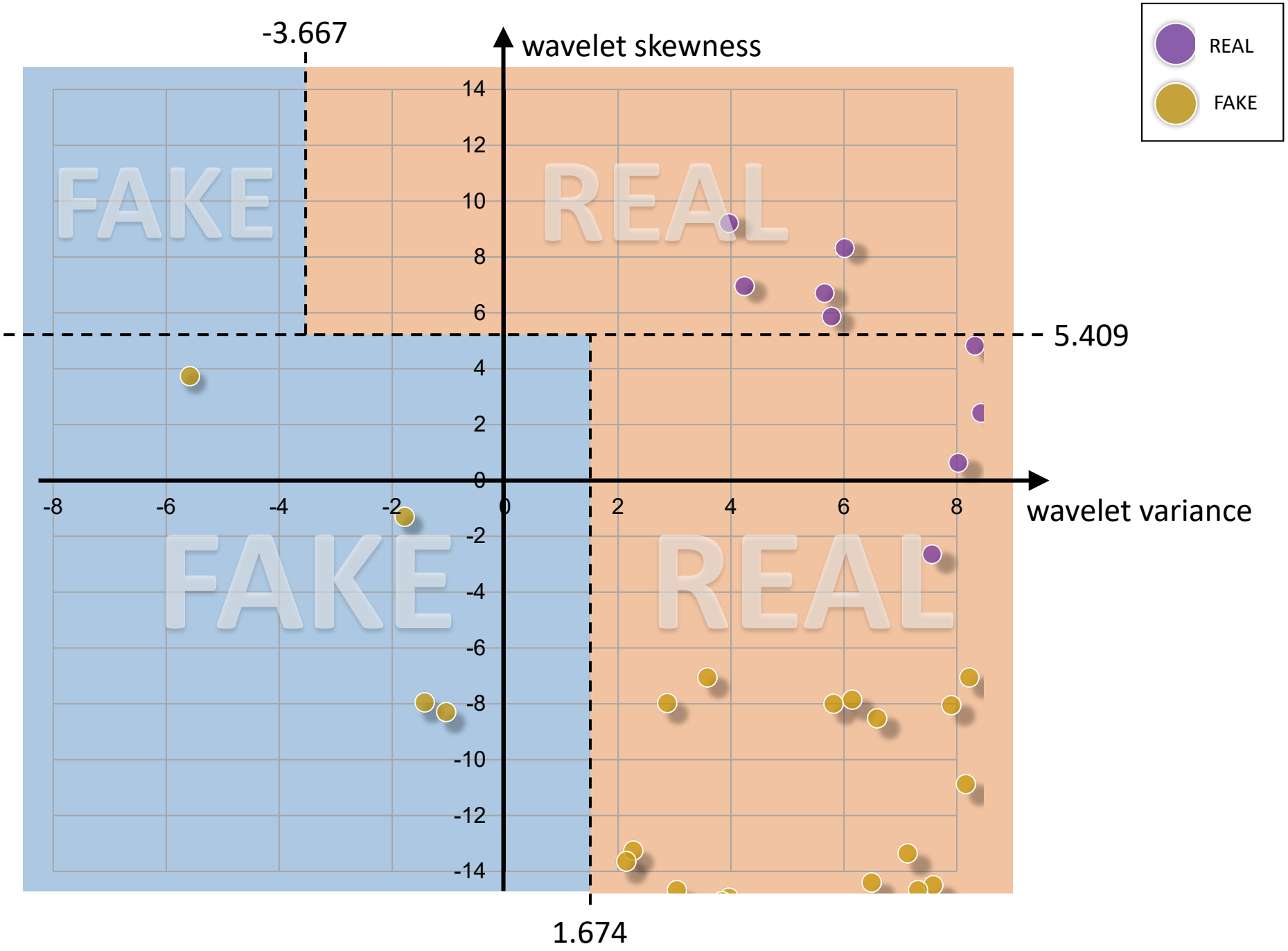
A decision tree trained on the 100 samples



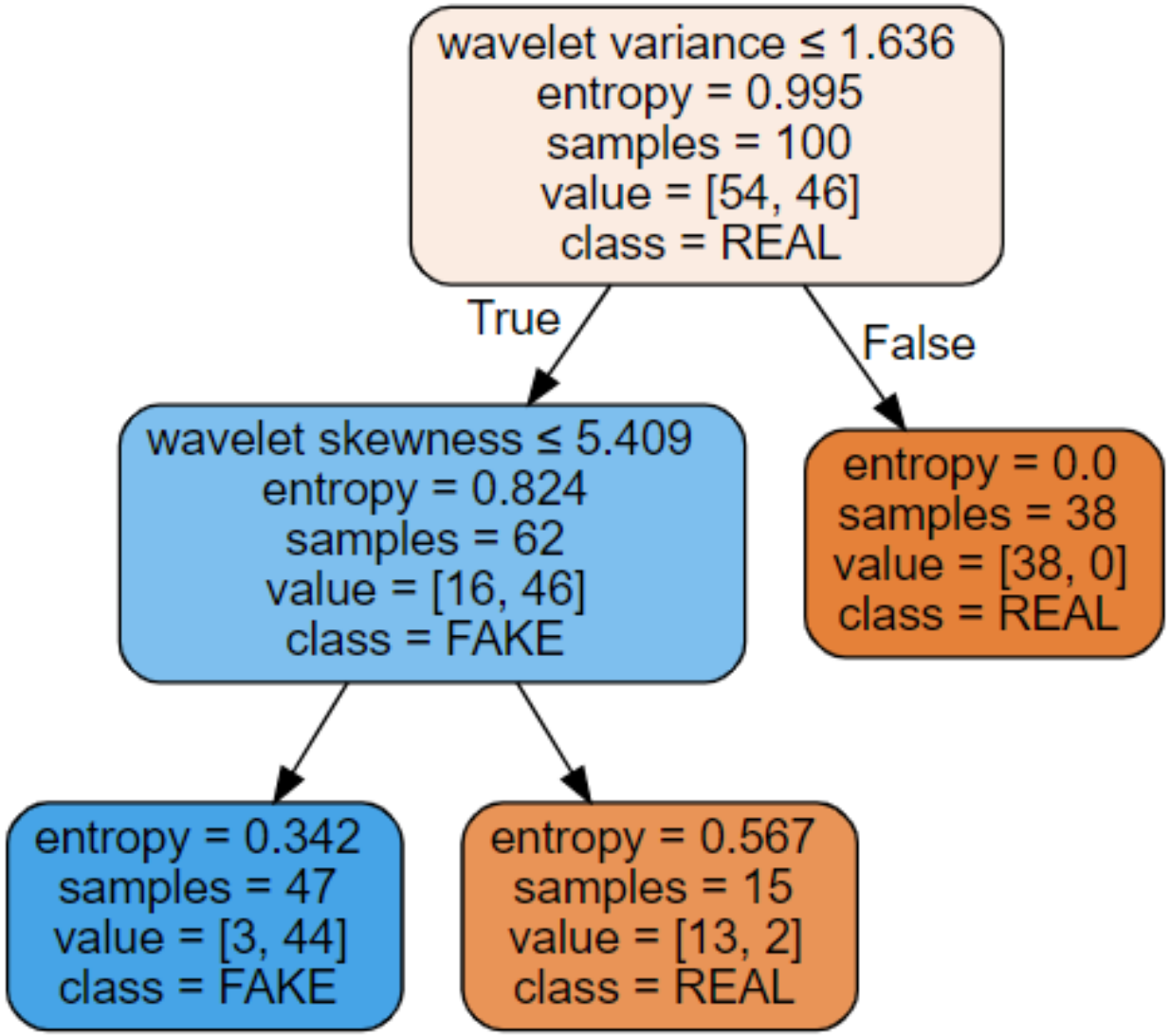
Accuracy = 97%


wavelet skewness
wavelet variance
5.409
REAL
FAKE
14
12
10
8
6
4
2
0
-2
-4
-6
-8
-10
-12
-14
-8
-6
-4
-2
0
2
4
6
8

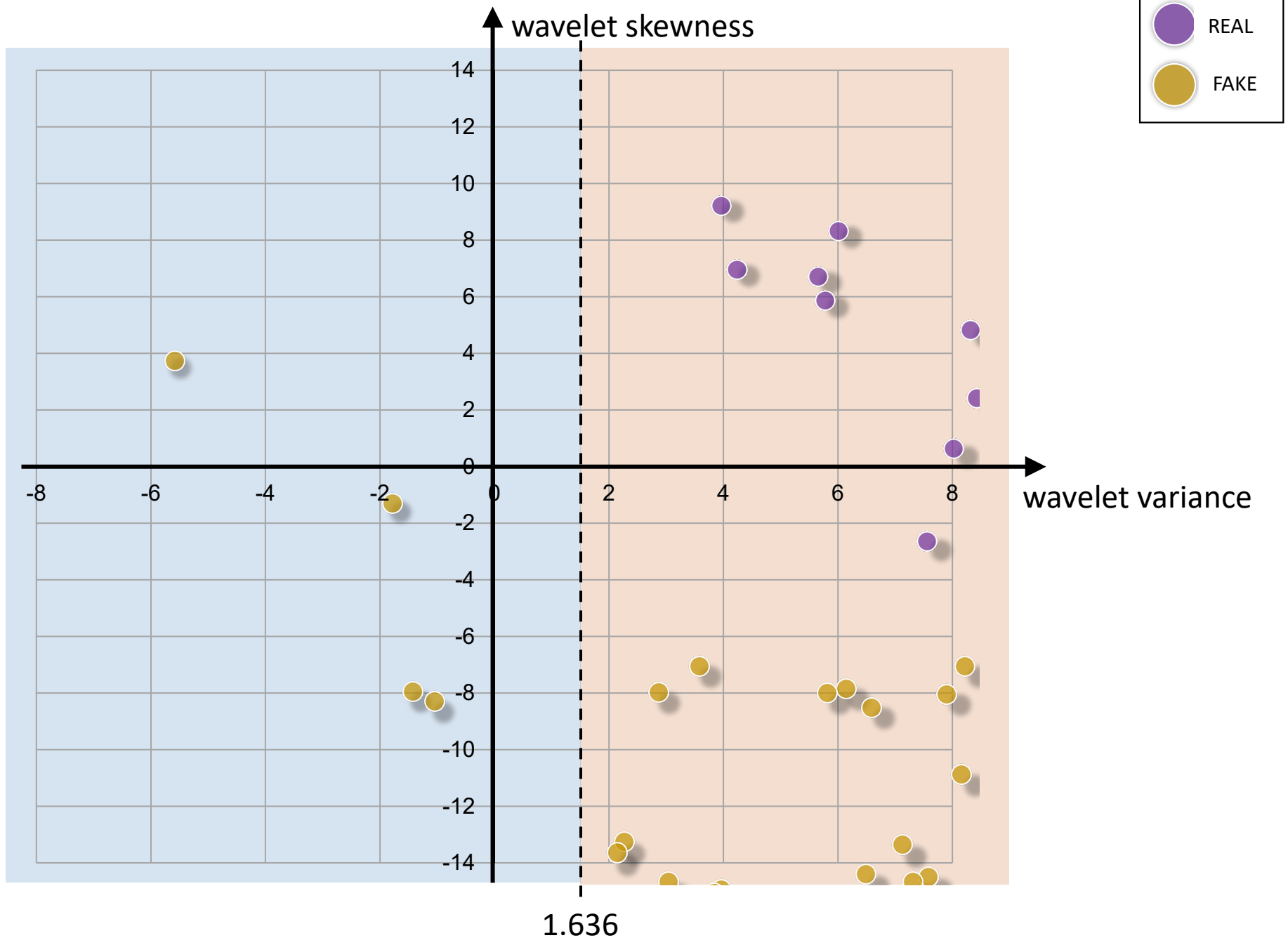

wavelet skewness
wavelet variance
REAL
FAKE
5.409
1.674
FAKE
REAL
14
12
10
8
6
4
2
0
-2
-4
-6
-8
-10
-12
-14
-8
-6
-4
-2
0
2
4
6
8

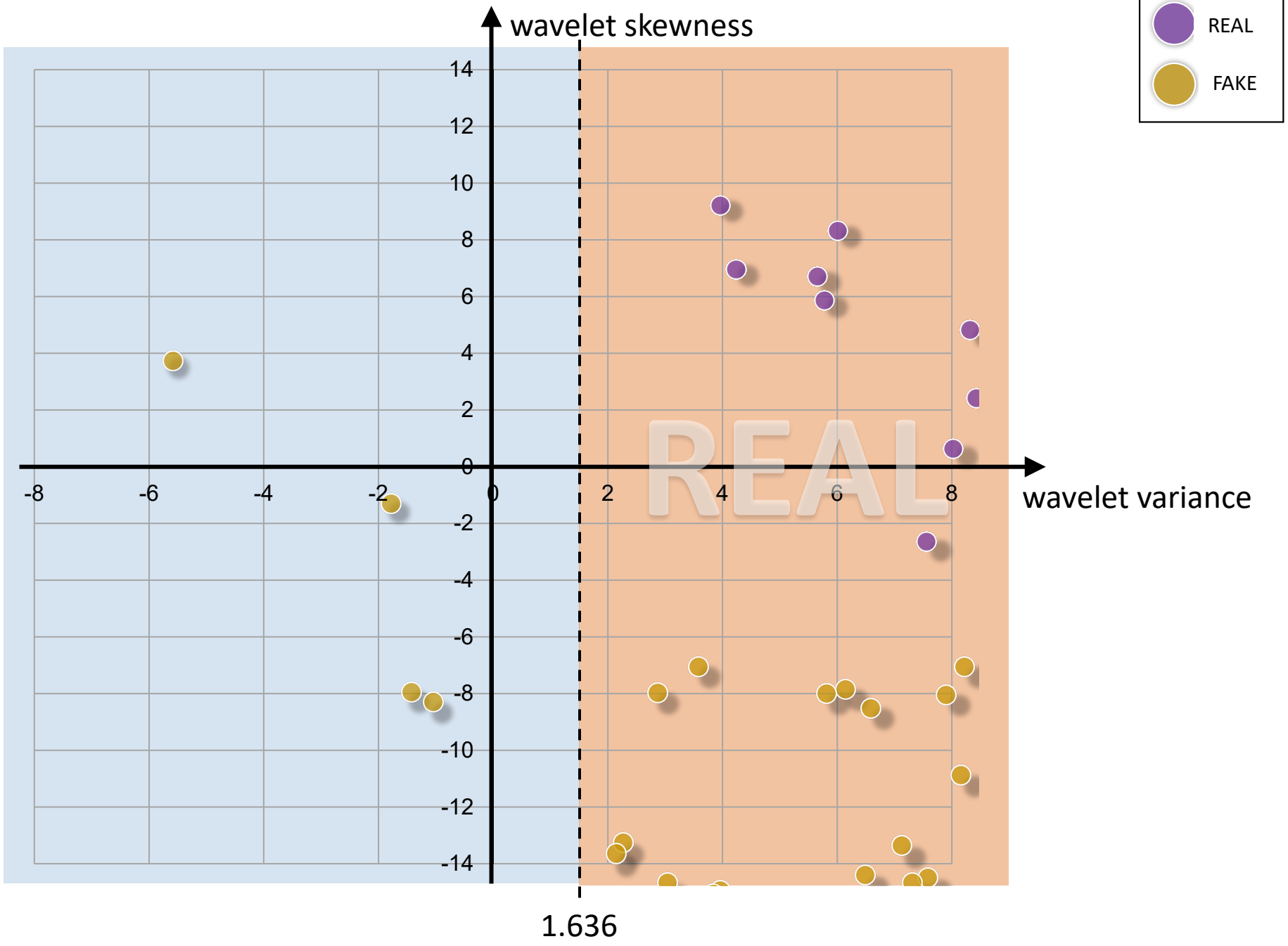

-3.667
wavelet skewness
REAL
FAKE
FAKE
REAL
FAKE
REAL
5.409
wavelet variance
1.674
14
12
10
8
6
4
2
0
-2
-4
-6
-8
-10
-12
-14
-8
-6
-4
-2
0
2
4
6
8

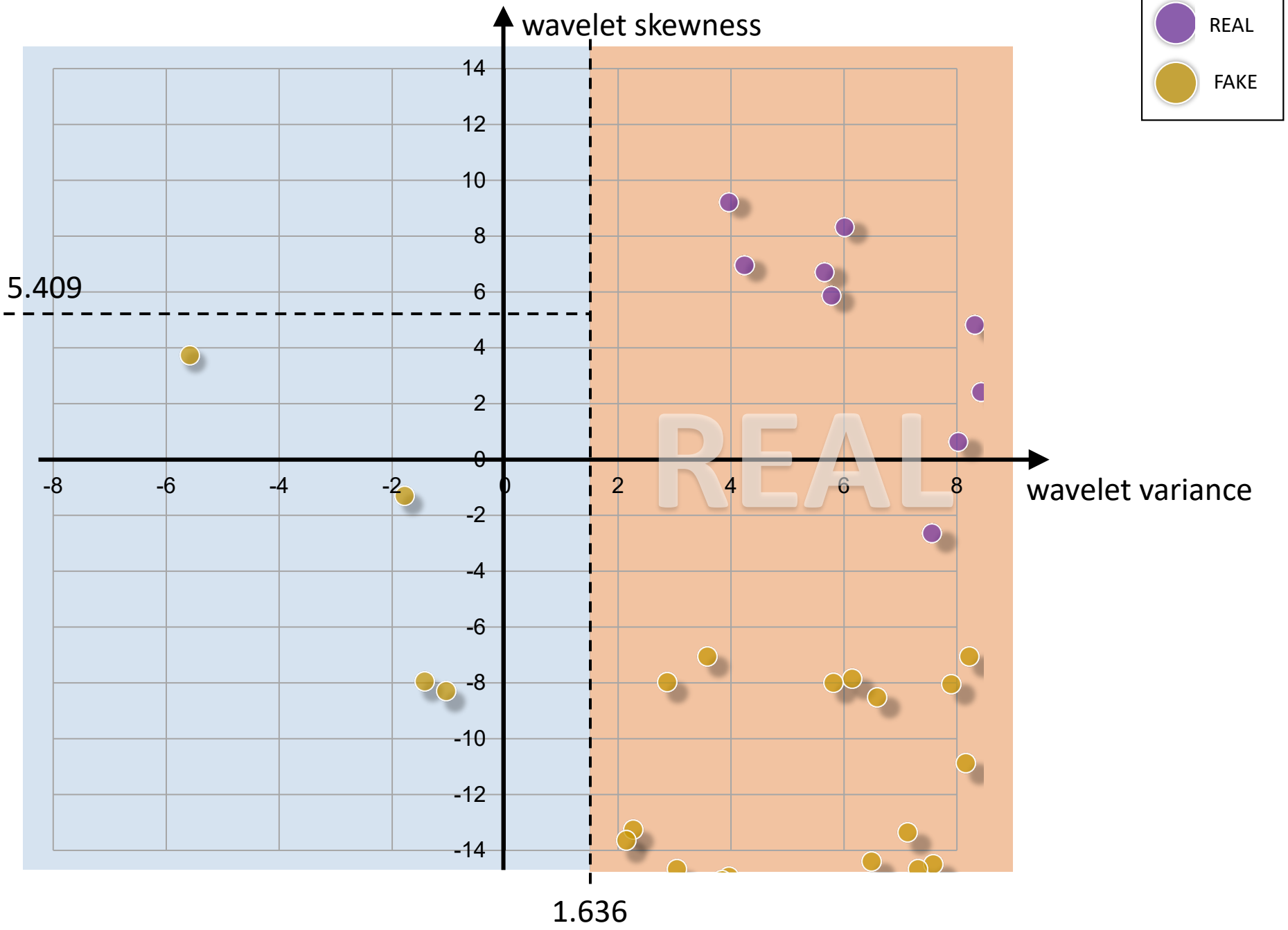
Another decision tree trained on the 100 samples

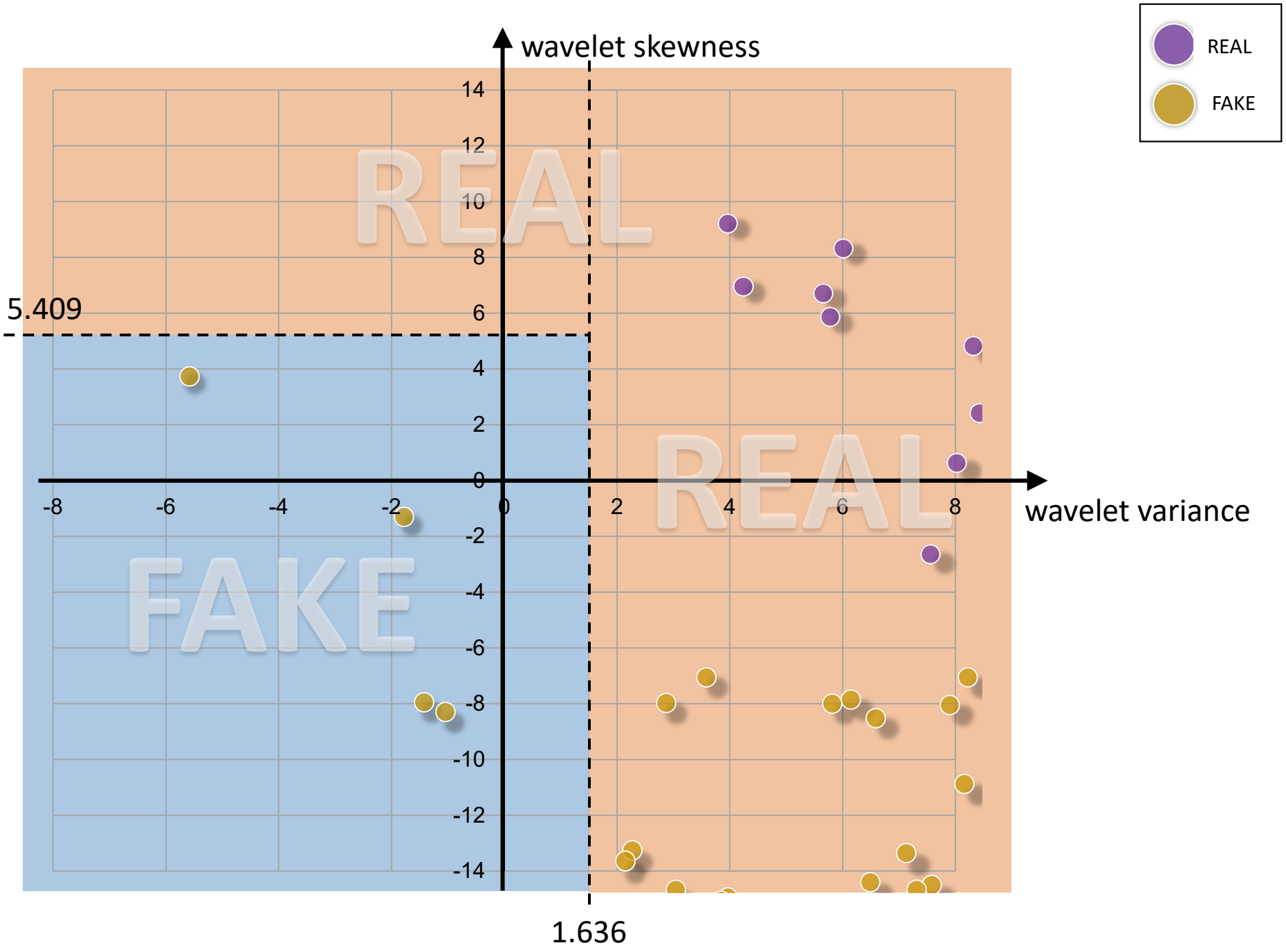


Accuracy = 95%


wavelet skewness
wavelet variance
1.636
REAL
FAKE
14
12
10
8
6
4
2
0
-2
-4
-6
-8
-10
-12
-14
-8
-6
-4
-2
0
2
4
6
8


wavelet skewness
wavelet variance
REAL
FAKE
REAL
14
12
10
8
6
4
2
0
-2
-4
-6
-8
-10
-12
-14
-8
-6
-4
-2
0
2
4
6
8
1.636

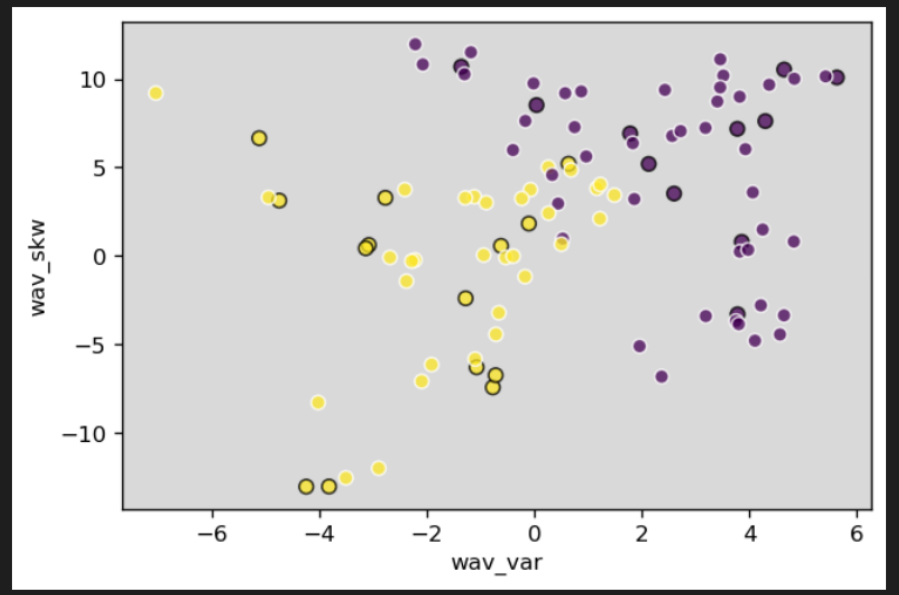

wavelet skewness
wavelet variance
REAL
FAKE
5.409
1.636
REAL
14
12
10
8
6
4
2
0
-2
-4
-6
-8
-10
-12
-14
-8
-6
-4
-2
0
2
4
6
8


wavelet skewness
wavelet variance
REAL
FAKE
5.409
1.636
REAL
REAL
FAKE
14
12
10
8
6
4
2
0
-2
-4
-6
-8
-10
-12
-14
-8
-6
-4
-2
0
2
4
6
8

Decision-tree classification on *banknote* data (2 features only)

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import graphviz
from sklearn.tree import export_graphviz
import numpy as np
import pandas as pd


#%% load the data
data = pd.read_excel("banknote.xlsx", sheet_name="mini",
  header=1, usecols="C:G")
data = data.drop(["wav_krt", "pix_ent"], axis=1)
data.head()


#%% split and plot the data
features = data.drop(["class"], axis=1)
X = np.array(features)
y = np.array(data["class"])

X_trn, X_tst, y_trn, y_tst = train_test_split(X, y,
  test_size=0.75, random_state=42)

ax = plt.subplot(111)
ax.set_facecolor((0.85, 0.85, 0.85))
ax.scatter(X_trn[:, 0], X_trn[:, 1], c=y_trn,
  edgecolors='k', alpha=0.75, s=40)
ax.scatter(X_tst[:, 0], X_tst[:, 1], c=y_tst,
  edgecolors='w', alpha=0.75, s=40)
plt.xlabel(features.columns[0])
plt.ylabel(features.columns[1])
plt.show()
```

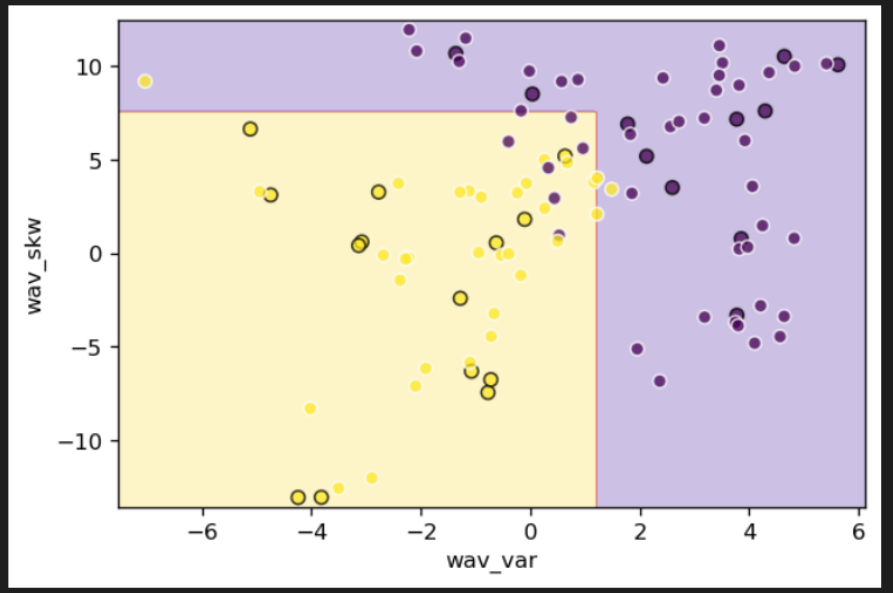
	wav_var	wav_skw	class
0	3.183600	7.23210	0
1	3.858400	0.78425	0
2	2.598900	3.51780	0
3	0.573400	9.19380	0
4	-0.016103	9.74840	0

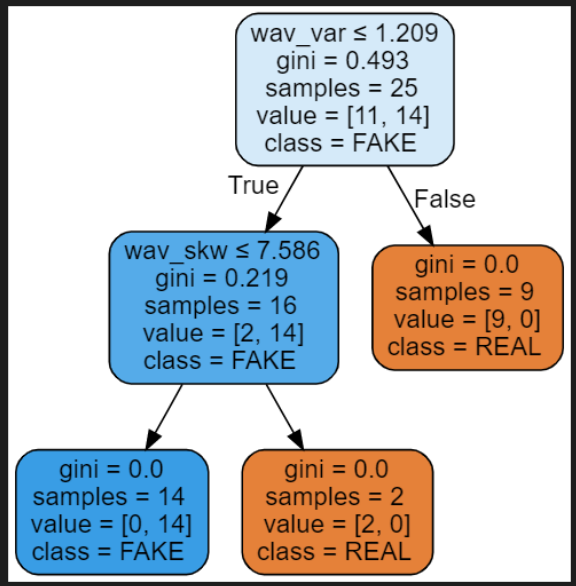


Decision-tree classification on *banknote* data (2 features only)

```
#%% fit and test the model
tree = DecisionTreeClassifier()
tree.fit(X_trn, y_trn)
y_trn_prd = tree.predict(X_trn)
print('Training accuracy: ',
  accuracy_score(y_true=y_trn, y_pred=y_trn_prd))
y_tst_prd = tree.predict(X_tst)
print('Testing accuracy: ',
  accuracy_score(y_true=y_tst, y_pred=y_tst_prd))

# %% plot the decision surface
x_min = X[:, 0].min() - 0.5; x_max = X[:, 0].max() + 0.5
y_min = X[:, 1].min() - 0.5; y_max = X[:, 1].max() + 0.5
xx, yy = np.meshgrid(
  np.arange(x_min, x_max, 0.02),
  np.arange(y_min, y_max, 0.02))

Z = tree.predict_proba(np.c_[xx.ravel(), yy.ravel()])
Z = Z.argmax(axis=1)
Z = Z.reshape(xx.shape)
ax = plt.subplot(111)
ax.contourf(xx, yy, Z, cmap="plasma", alpha=0.25)

ax.scatter(X_trn[:, 0], X_trn[:, 1], c=y_trn,
  edgecolors="k", alpha=0.75)
ax.scatter(X_tst[:, 0], X_tst[:, 1], c=y_tst,
  edgecolors="w", alpha=0.75)
plt.xlabel(features.columns[0])
plt.ylabel(features.columns[1])

#%% show the tree
dot_data = export_graphviz(tree, out_file=None,
  feature_names=features.columns, class_names=["REAL", "FAKE"],
  filled=True, rounded=True, special_characters=True)
graph = graphviz.Source(dot_data)
graph
```

```
Training accuracy:  1.0
Testing accuracy:  0.8666666666666667
```





Today: Classification and Regression, Part 1b

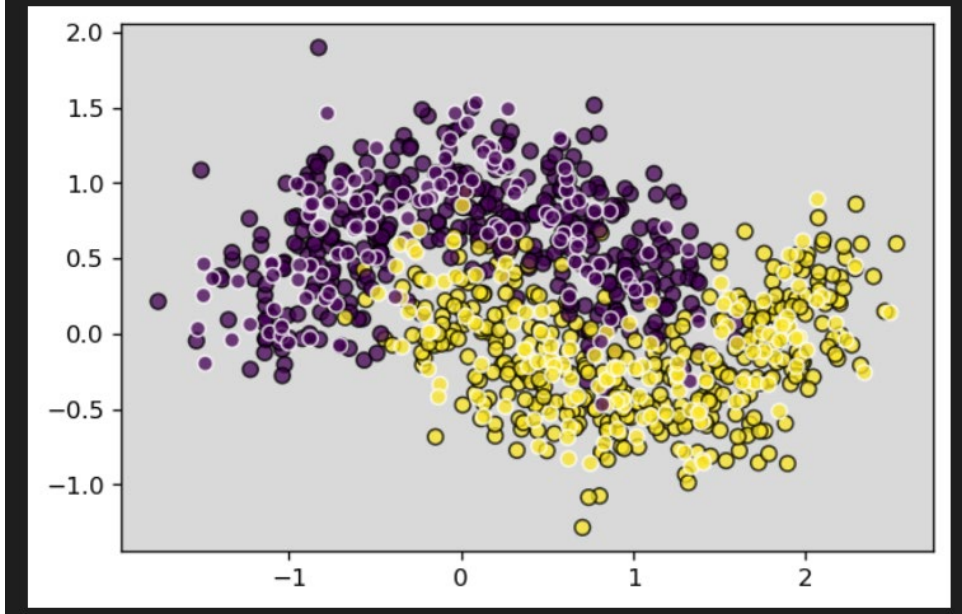
1. Bayesian classification of another dataset
2. Decision trees (classification)
3. Random forests (classification)

Random Forests

- **Decision trees** tend to **overfit** the training data
- Idea: Use **multiple trees** (ensemble of trees)
  - Use majority rule to make final class decision
  - Train **many trees** on **subsets** of the training **data**
    - Bootstrapping + aggregating = "bagging"
    - If the trees are not correlated, then ensemble can yield better classification
    - But, all trees influenced by most dominant feature(s) → correlation
  - Train **many trees** on **subsets** of the **features**
    - Random subspace method, a.k.a. "Feature bagging"
    - Tends to yield less correlated trees → better classification
  - ***Random forest*** classifier uses these approaches
    - Uses "feature bagging" via
      - Search for each split point is limited to a random subset of features
    - Can also use training data "bagging"

Random forest classification on *moons* data

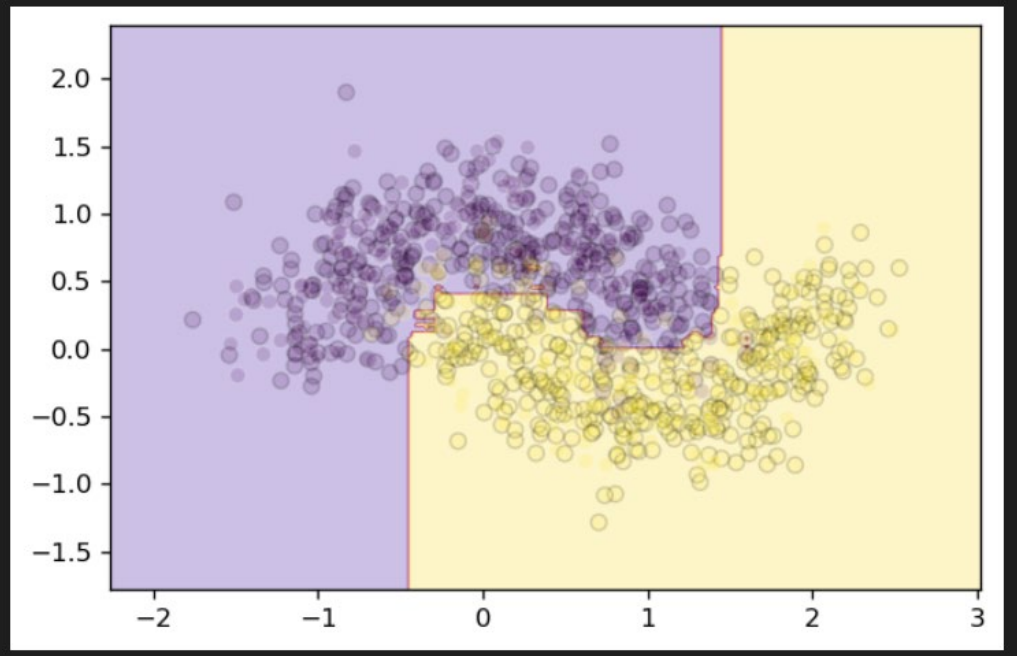
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.datasets import make_moons
import numpy as np
import pandas as pd

#%% create, split, and plot the data
X, y = make_moons(n_samples=1000, noise=0.25, random_state=1)

X_trn, X_tst, y_trn, y_tst = train_test_split(X, y,
  test_size=0.25, random_state=42)

ax = plt.subplot(111)
ax.set_facecolor((0.85, 0.85, 0.85))
ax.scatter(X_trn[:, 0], X_trn[:, 1], c=y_trn,
  edgecolors='k', alpha=0.75, s=40)
ax.scatter(X_tst[:, 0], X_tst[:, 1], c=y_tst,
  edgecolors='w', alpha=0.75, s=40)
plt.show()

#%% test the model
forest = RandomForestClassifier(n_estimators=500, max_depth=7)
forest.fit(X_trn, y_trn)

y_trn_prd = forest.predict(X_trn)
print('Training accuracy: ',
  accuracy_score(y_true=y_trn, y_pred=y_trn_prd))
y_tst_prd = forest.predict(X_tst)
print('Testing accuracy: ',
  accuracy_score(y_true=y_tst, y_pred=y_tst_prd))
```




```
Training accuracy:  0.9666666666666667
Testing accuracy:  0.932
```

Random forest classification on *moons* data

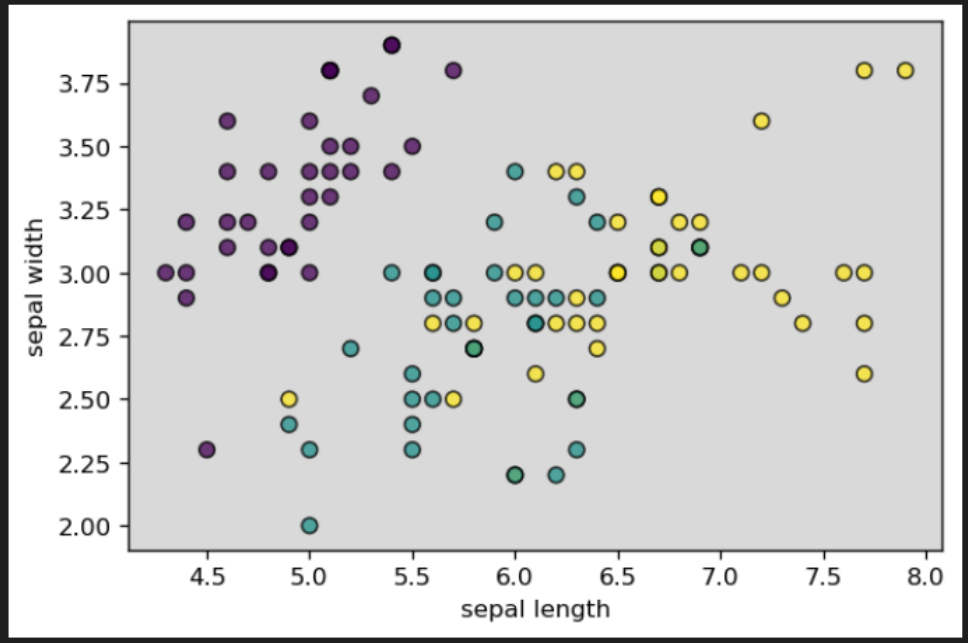
```
# %% plot the decision regions
x_min = X[:, 0].min() - 0.5
x_max = X[:, 0].max() + 0.5
y_min = X[:, 1].min() - 0.5
y_max = X[:, 1].max() + 0.5
xx, yy = np.meshgrid(
  np.arange(x_min, x_max, 0.02),
  np.arange(y_min, y_max, 0.02)
  )

Z = forest.predict_proba(np.c_[xx.ravel(), yy.ravel()])
Z = Z.argmax(axis=1)
Z = Z.reshape(xx.shape)
ax = plt.subplot(111)
ax.contourf(xx, yy, Z, cmap="plasma", alpha=0.25)

ax.scatter(X_trn[:, 0], X_trn[:, 1], c=y_trn,
  edgecolors="k", alpha=0.15)
ax.scatter(X_tst[:, 0], X_tst[:, 1], c=y_tst,
  edgecolors="w", alpha=0.15)

plt.show()
```



Random forest classification on *iris* data (2 features only)

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import numpy as np
import pandas as pd

#%% load, split, and plot the data
data = pd.read_excel("iris.mini.xlsx",
  sheet_name="iris (all)", usecols="F:J", header=1)
data = data.drop(["petal length", "petal width"], axis=1)
data.head()

features = data.drop(["class"], axis=1)
X = np.array(features)
y = np.array(data["class"])

X_trn, X_tst, y_trn, y_tst = train_test_split(X, y,
  test_size=0.25, random_state=42)

c_dict = {"Iris-setosa": 0, "Iris-versicolor": 0.5,
  "Iris-virginica": 1}
clrs_y = np.zeros(y_trn.shape)
for idx in range(len(y_trn)):
  clrs_y[idx] = c_dict[y_trn[idx]]

ax = plt.subplot(111)
ax.set_facecolor((0.85, 0.85, 0.85))
ax.scatter(X_trn[:, 0], X_trn[:, 1], c=clrs_y,
  edgecolors='k', alpha=0.75, s=40)
plt.xlabel(features.columns[0])
plt.ylabel(features.columns[1])
plt.show()
```

	sepal length	sepal width	class
0	5.8	2.7	Iris-virginica
1	4.6	3.4	Iris-setosa
2	6.3	2.3	Iris-versicolor
3	6.8	3.2	Iris-virginica
4	6.2	2.9	Iris-versicolor



Random forest classification on *iris* data (2 features only)

```
#%% train the model
forest = RandomForestClassifier(n_estimators=1000,
  max_depth=7, min_samples_leaf=3)
forest.fit(X_trn, y_trn)

y_trn_prd = forest.predict(X_trn)
print('Training accuracy: ',
  accuracy_score(y_true=y_trn, y_pred=y_trn_prd))
y_tst_prd = forest.predict(X_tst)
print('Testing accuracy: ',
  accuracy_score(y_true=y_tst, y_pred=y_tst_prd))

#%% plot the decision regions
c_dict = {"Iris-setosa": 0, "Iris-versicolor": 0.5,
  "Iris-virginica": 1}
clrs_y = np.zeros(y_trn.shape)
for idx in range(len(y_trn)):
  clrs_y[idx] = c_dict[y_trn[idx]]

x_min = X[:, 0].min() - 0.5; x_max = X[:, 0].max() + 0.5
y_min = X[:, 1].min() - 0.5; y_max = X[:, 1].max() + 0.5
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
  np.arange(y_min, y_max, 0.02))

Z = forest.predict_proba(np.c_[xx.ravel(), yy.ravel()])
Z = Z.argmax(axis=1)
Z = Z.reshape(xx.shape)
ax = plt.subplot(111)
ax.contourf(xx, yy, Z, cmap="plasma", alpha=0.25)

ax.scatter(X_trn[:, 0], X_trn[:, 1], c=clrs_y,
  edgecolors="k", alpha=0.75)
plt.xlabel(features.columns[0])
plt.ylabel(features.columns[1])
```

```
Training accuracy:  0.8571428571428571
Testing accuracy:  0.7105263157894737
```