

問1（ワインデータ）

- (1) 隠れ素子数の数 M や正則化項のパラメータ λ は、CV法を用いて汎化誤差が最小になるように選択した。具体的には、 M は 1~10, λ は 0.1, 0.01, 0.001, 0.0001, 0.00001 でそれぞれを組み合わせて調査した。選択した結果は以下のようになつた。プログラムはソースコード 1 であり、この時の CV 法は 10 分割で行い、それによる誤り率は、0.064 であった。また、正確度は 0.98 であった。

$$M = 6, \lambda = 0.00001$$

図 1 のように識別境界のプロットを試みたが、上手くプロットすることができなかつた。

- (2) 線形カーネルと RBF カーネルを取り上げ、for 文でいくつかのパラメータ c や σ で調査した結果、汎化誤差が最小になつたのは、線形カーネルを用いた時の $c = 1$ であった。この時の正確度は、1.0 であった。（RBF カーネルよりも線形カーネルの方が正確度が高かつた。）また、識別境界のプロットを以下の図 1 に示す。このプロットは、最初に LDA(Linear Discriminant Analysis) を使って 13 次元のワインデータを 2 次元に射影した。

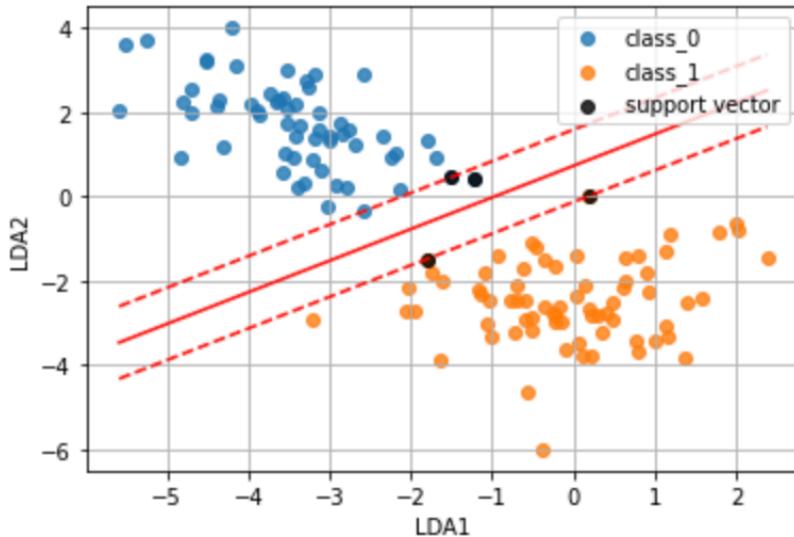


図 1: SVM の識別境界のプロット（線形カーネル, $c = 1$ ）

- (3) (1), (2) の結果、ワインデータの識別には、サポートベクトルマシンの学習規則を用いる方が望ましい。

正確度を見ると、サポートベクトルマシンの方が大きいため望ましいと考える。これは、データの次元が大きい場合でも識別精度が良いという SVM の特徴が現れているからだと考える。しかし、誤差逆伝搬法の学習規則でも正確度が高いと言えるため、どちらの学習規則も精度は良いと考えられる。

問2（カニデータ）

- (1) 体長を記録した 5 つの特徴を用いて主成分分析を行い、2 次元に射影したデータのプロットは以下の図 2 である。また以下、PC1, PC2 を第 1 主成分、第 2 主成分として表している。

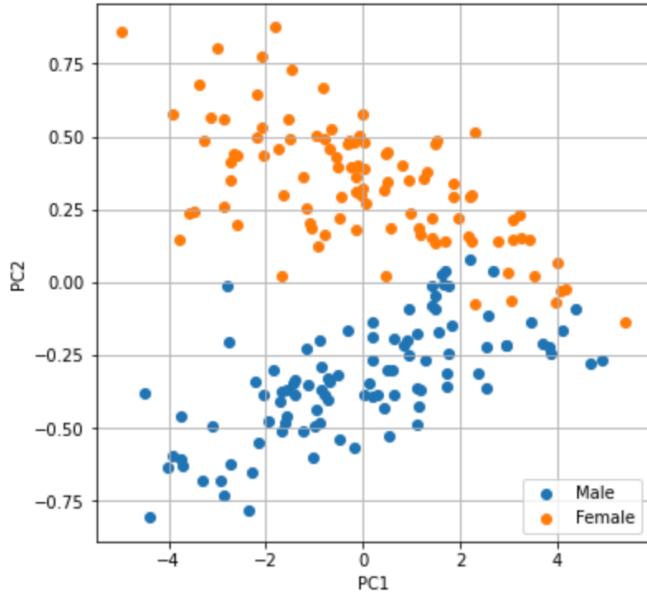


図 2: 5つの特徴の主成分分析による2次元に射影したデータのプロット

また, 寄与率と累積寄与率は以下の表 1 の結果となった. また, 累積寄与率のプロットを以下の図 3 に示す.

表 1: カニデータの寄与率と累積寄与率

	PC1	PC2	PC3	PC4	PC5
寄与率	0.958	0.0303	0.00933	0.00223	0.000342
累積寄与率	0.958	0.988	0.998	1.00	1.00

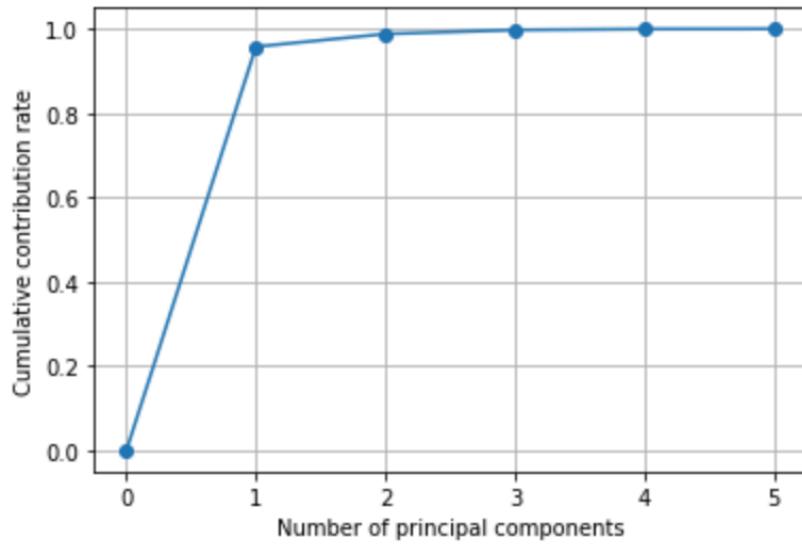


図 3: カニデータの累積寄与率のプロット

従って, 第1主成分の寄与率は, 0.958, 第2主成分の寄与率は, 0.0303である. 累積寄与率は, 表 1 の通りである.

(2) 体長を記録した5つの特徴を用いて階層的クラスタリングを行った結果は、以下の図4のデンドログラム通りである。

手法については、クラスタ間の距離は、Ward法、対象間の距離は、ユークリッド距離を採用した。

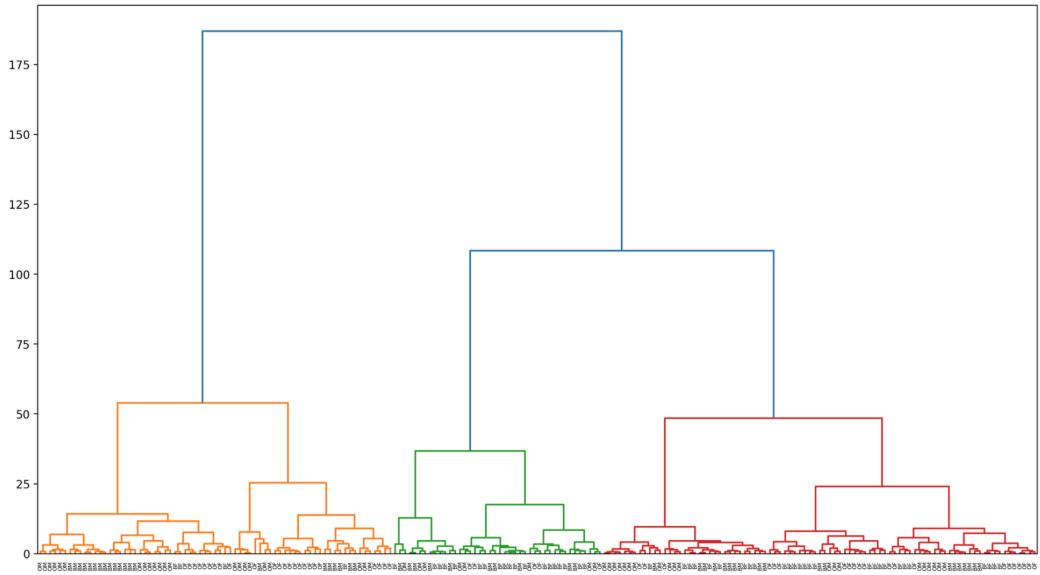


図4: カニデータのクラスタリング結果 (Ward法)

また、手法を対象間の距離は、ユークリッド距離のままで、クラスタ間の距離を最短距離法 (Single) にした場合の結果を以下の図5に示す。

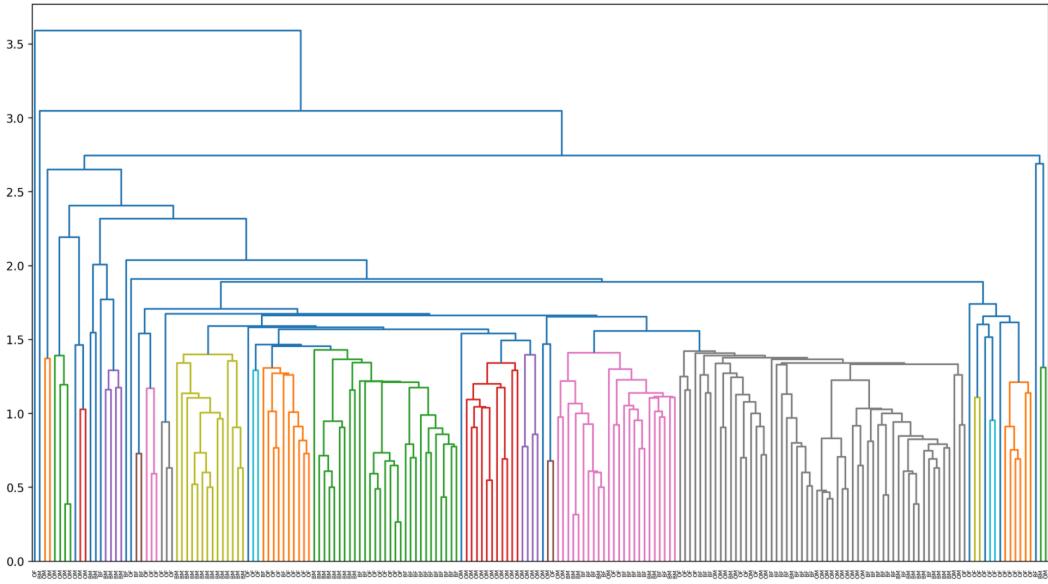


図5: カニデータのクラスタリング結果 (最短距離法)

教師なし分類とは、教師が無い（予め与えられたクラスが無い）学習によって分類するということであり、この場合、入力データ間の距離や類似度、統計的な性質に基づいて、ク

ラスを自動的に生成するということである。そして、今回行った「階層的クラスタリング」が教師なし分類である。

図4の階層的クラスタリングは、比較的グループ分けができていると考える。それに対し図5では、鎖のように連なっているようになっており、非常に分かりづらいグループ分けとなっている。よって、階層的クラスタリングは、対象間とクラスタ間の距離の定義が非常に重要であることが言える。また、クラスタリングは距離にのみ基づいているため、解釈するためには、別的情報が必要になってくると考えられる。

- (3) 問1と同様に、誤差逆伝搬法とサポートベクトルマシンを用いた識別を行う。

誤差逆伝搬法では、CV法を用いて汎化誤差が最小になるように M と λ を選択した。その結果、以下のようにになった。

$$M = 7, \lambda = 0.1$$

この時の正確度は、0.983 であった。

SVMでも問1と同様に（ソースコード1と同様のパラメータの調べ方）、調べた結果、線形カーネルとRBFカーネル共に正確度が0.983 になった。この時のパラメータは、以下の表2の通りである。

表2: SVMのパラメータ

	線形カーネル	RBFカーネル
パラメータ	$c = 10$	$c = 1000, \sigma = 0.8$

SVMの識別境界のプロットを以下の図6に示す。

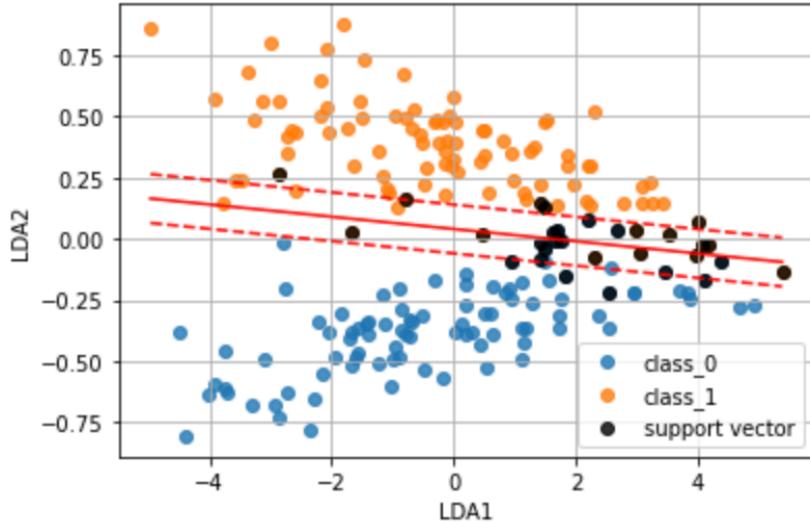


図6: SVMの識別境界のプロット（線形カーネル, $c = 10$ ）

以上より、誤差逆伝搬法とSVMの両者の正確度は同じ結果となり、どちらも精度の良い識別規則であることが言える。

問3 (タイタニックデータ)

(1) 今回用いる変数は, "Pclass", "Age", "Sex", "Fare", "SibSp", "Parch", "Embarked" の7つの変数である。(プログラムで容易に扱えると考えた7つである。)

そして決定木, バギング, ブースティング(アダブースト), ランダムフォレストにより識別器を構成し, 学習データを交差検証法によって誤り率を求め, 以下の表3にまとめた。交差検証法は全て10分割に設定している。

表3: 学習データの誤り率

	決定木	バギング	アダブースト	ランダムフォレスト
誤り率	0.184	0.173	0.186	0.165

また, 木の深さや集団学習に使う弱学習器の個数を変化させながらの正確度のプロットは, 以下の図7, 図8, 図9, 図10に示す。

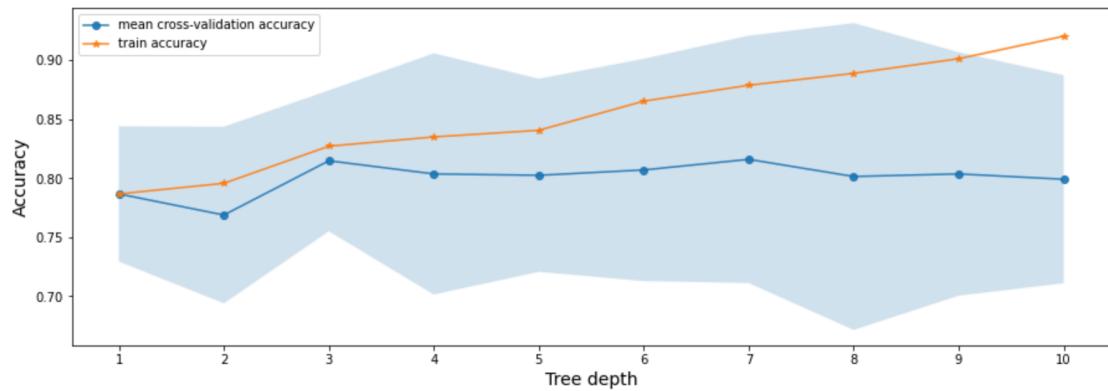


図7: 決定木の正確度のプロット

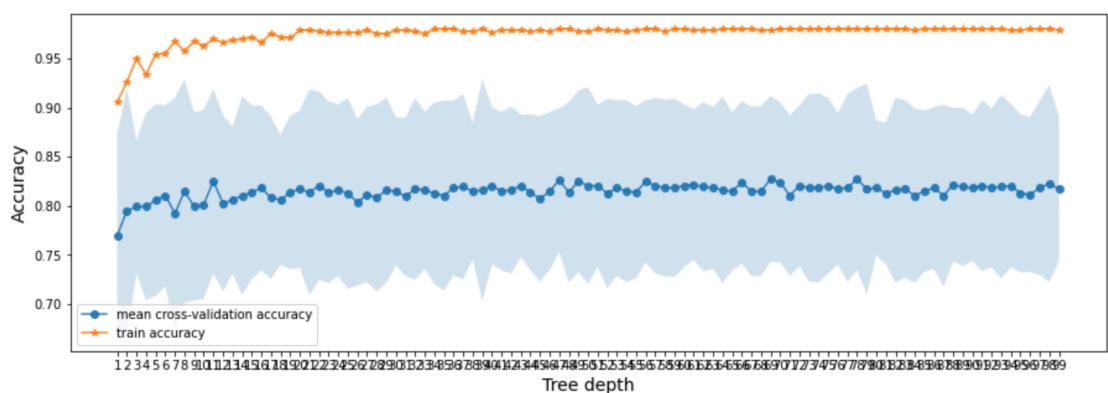


図8: バギングの正確度のプロット

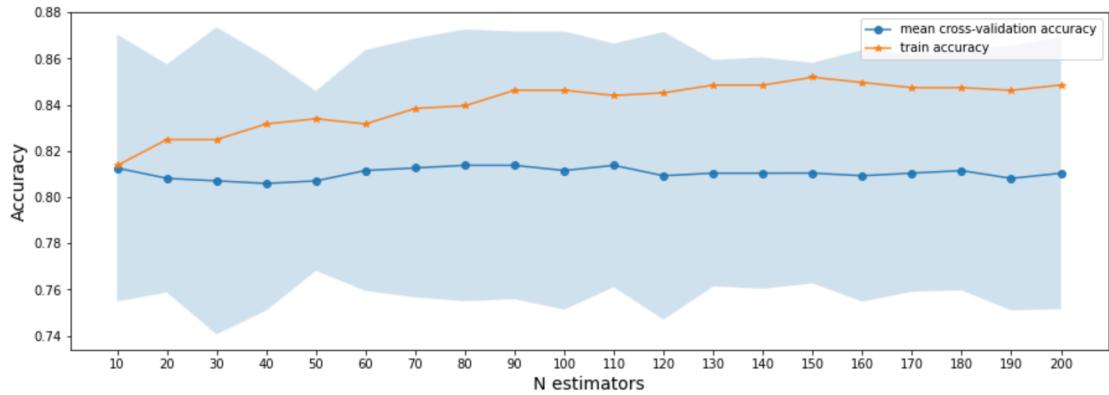


図 9: アダブーストの正確度のプロット

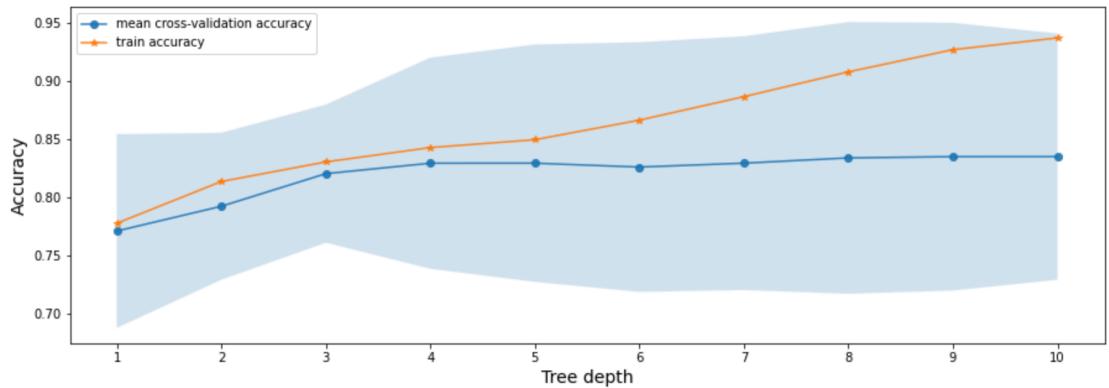


図 10: ランダムフォレストの正確度のプロット

上記の結果より、ランダムフォレストが4つの中で最も精度の良い識別器であることが分かった。

次にテストデータの誤り率であるが、test データには Survived 生存フラグが無いため、今回は、train データの Survived 生存フラグと test データを用いて、交差検証法による誤り率を求めることを試みた。誤り率の結果は以下の表 4 に示す。

表 4: テストデータの誤り率

	決定木	バギング	アダブースト	ランダムフォレスト
誤り率	0.390	0.411	0.402	0.390

正しい識別では無い可能性があるが、こちらでもランダムフォレストの精度が比較的良いということが分かる。また、ランダムフォレストは森のサイズによる過学習が発生しないことが分かっているため、さらにサイズを大きくさせれば、精度が良くなると考えられる。

- (2) ランダムフォレスト（木の深さの最大値は4とした。）を用いて、重要変数のプロットを行った結果を以下の図 11 に示す。（train データを用いた。）

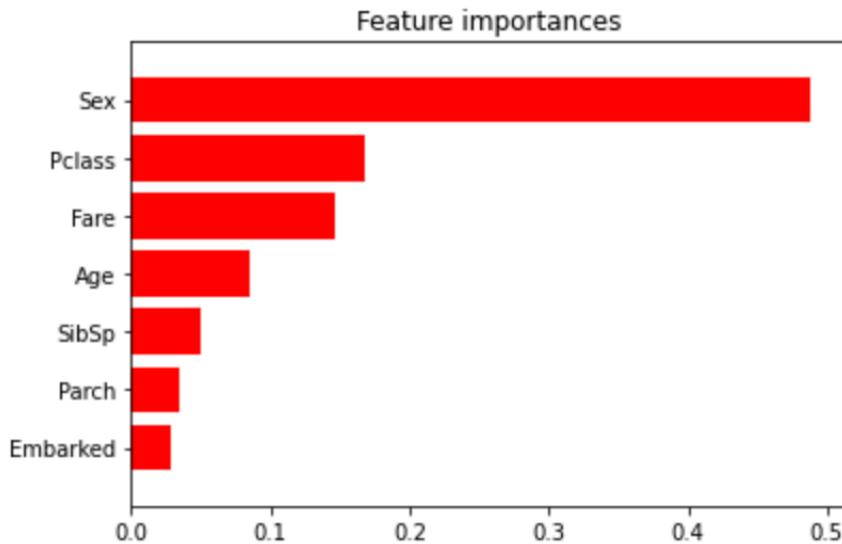


図 11: ランダムフォレストによる重要変数のプロット

図11より、ランダムフォレストでは識別に最も寄与する重要変数は、"Sex"で、次に" Pclass"であることが分かった。

同様に、ブースティング（アダブーストで n_estimators=200）を用いて、重要変数のプロットを行った結果を以下の図 12 に示す。（train データを用いた。）

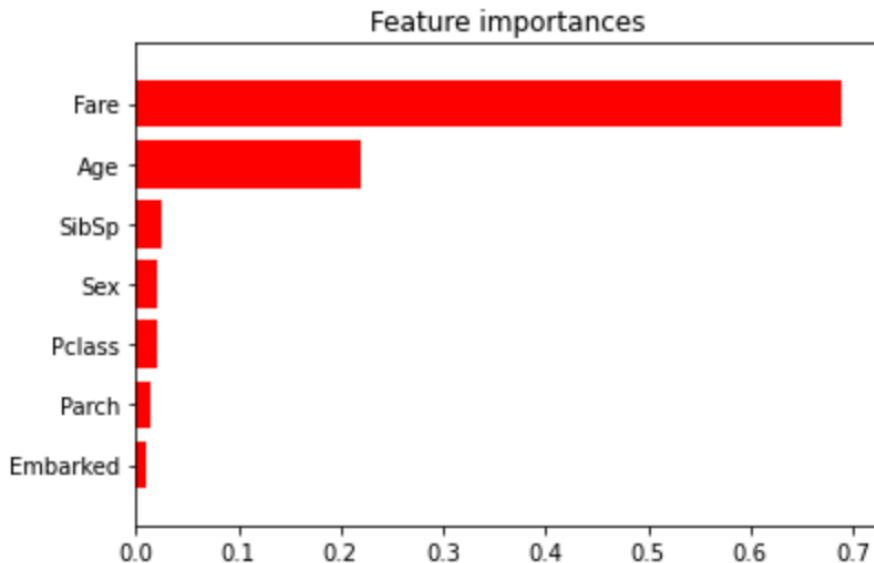


図 12: アダブーストによる重要変数のプロット

図12より、アダブーストでは識別に最も寄与する重要変数は、"Fare"で、次に"Age"であることが分かった。

ランダムフォレストとアダブーストで重要変数の結果は異なるが、両者とも考慮すると、最も寄与している変数は、図 11、図 12 より、"Fare"であると考える。（ランダムフォレストでは、3 番目に、アダブーストでは、1 番目に寄与しているという結果であるため。）

参考文献

- [1] 主成分分析を Python で理解する
URL : <https://qiita.com/maskot1977/items/082557fcda78c4cdb41f>
最終閲覧日 : 2020 年 9 月 1 日
- [2] 【Kaggle 初心者入門編】タイタニック号で生き残るのは誰?
URL : <https://www.codexa.net/kaggle-titanic-beginner/>
最終閲覧日 : 2020 年 9 月 1 日
- [3] クラスタリング手法のクラスタリング
URL : <https://qiita.com/suecharo/items/20bad5f0bb2079257568>
最終閲覧日 : 2020 年 9 月 1 日

付録

ソースコード 1: 問 1 のパラメータなどのチューニング

```
1 from sklearn.datasets import load_wine
2 from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as lda
3 wine = load_wine()
4
5 import pandas as pd
6 pd.DataFrame(wine.data, columns=wine.feature_names)
7
8 X = wine.data
9 y = wine.target
10 target_names = wine.target_names
11
12 X_tr, X_te, Y_tr, Y_te = train_test_split(X, y, test_size=0.3, random_state
13 =2020)
14
15 from sklearn.model_selection import train_test_split
16 from sklearn.metrics import accuracy_score, f1_score
17 from sklearn.neural_network import MLPClassifier
18 import numpy as np
19
20 # 誤差逆伝搬法の学習規則 M λ
21 cv_error = 100
22 alpha = [0.1, 0.01, 0.001, 0.0001, 0.00001]
23 cv = 10
24 scoring='accuracy'
25
26 cv_scores_list = []
27 cv_scores_std = []
28 cv_scores_mean = []
29 accuracy_scores = []
30
31 for i in range(1, 11):
32     print("i = ", i)
```

```

33     for j in alpha:
34         clf = MLPClassifier(hidden_layer_sizes=(i, ), alpha=j,
35                             activation ='logistic',
36                             solver="adam", random_state=2020,
37                             max_iter=10000)
38         clf_fit = clf.fit(X_tr, Y_tr)
39         Y_pred = clf.predict(X_te)
40         cv_scores = cross_val_score(clf, X_tr, Y_tr, cv=cv, scoring=scoring)
41         cv_scores_list.append(cv_scores)
42         cv_scores_mean.append(cv_scores.mean())
43         cv_scores_std.append(cv_scores.std())
44         accuracy_scores.append(accuracy_score(y_true=Y_te, y_pred=Y_pred))
45         if cv_error > 1-np.mean(cv_scores):
46             cv_error = 1-np.mean(cv_scores)
47             cv_score_mean = cv_scores.mean()
48             cv_score_std = cv_scores.std()
49             error_min = 1-accuracy_score(y_true=Y_te, y_pred=Y_pred)
50             best_accuracy = accuracy_score(y_true=Y_te, y_pred=Y_pred)
51             alpha_min = j
52             hidden_min = i
53
54 print("accuracy = ", max(accuracy_scores))
55 print("正確度 = ", best_accuracy)
56 print("cv error = ", cv_error)
57 print("汎化誤差 = ", error_min)
58 print("hidden_layer_sizes = ", hidden_min)
59 print("alpha = ", alpha_min)
60
61
62 # サポートベクトルマシン
63 from sklearn.svm import SVC
64 import numpy as np
65
66
67 C = [0.01, 0.1, 1, 10, 100, 1000, 10000]
68
69 # 線形カーネル
70 error_min = 100
71 for c in C:
72     model = SVC(C=c, kernel='linear')
73     model.fit(X_tr, Y_tr)
74     Y_pred = model.predict(X_te)
75     if error_min > 1-accuracy_score(Y_te, Y_pred):
76         error_min = 1-accuracy_score(Y_te, Y_pred)
77         best_accuracy = accuracy_score(y_true=Y_te, y_pred=Y_pred)
78         best_C = c
79         best_y_pred = Y_pred
80
81 print("*** Best ***\n (線形カーネル)\nC:%s" %(best_C))
82 print('正確度: ', best_accuracy)
83 print('汎化誤差: ', error_min)
84
85

```

```

86 # RBF カーネル
87 error_min = 100
88 sigma= np.array([0.2,0.4,0.8,2,4])
89 sigma = 1/((sigma**2)*2)
90
91 for c in C:
92     for sig in sigma:
93         model= SVC(C=c, kernel='rbf', gamma=sig)
94         model.fit(X_tr, Y_tr)
95         Y_pred = model.predict(X_te)
96         if error_min > 1-accuracy_score(Y_te, Y_pred):
97             error_min = 1-accuracy_score(Y_te, Y_pred)
98             best_accuracy = accuracy_score(y_true=Y_te, y_pred=Y_pred)
99             best_sigma = sig
100            best_C = c
101            best_y_pred = Y_pred
102
103 print("\n(RBF カーネル)\n C:%s, gamma:%s" %(best_C, best_sigma))
104 print('正確度: ', best_accuracy)
105 print('汎化誤差: ', error_min)

```

ソースコード 2: 問 1, 問 2 の SVM の識別境界

```

1 # 2次元に縮小
2 lda = lda(n_components=2)
3 Z = lda.fit(X, y).transform(X)
4 Z1 = Z[y == 0, ]
5 Z2 = Z[y == 1, ]
6 Z = np.concatenate([Z1, Z2], axis=0)
7
8 from sklearn.svm import SVC
9 clf= SVC(C=1, kernel='linear')
10 clf.fit(Z, Y)
11
12 from matplotlib.colors import ListedColormap
13 colors = ['blue', 'red', 'green']
14
15 for color, i, target_name in zip(colors, [0, 1], target_names):
16     plt.scatter(Z[Y == i, 0], Z[Y == i, 1], alpha=.8,
17                  label=target_name)
18 plt.xlabel('LDA1')
19 plt.ylabel('LDA2')
20 plt.scatter(Z[clf.support_, 0], Z[clf.support_, 1], alpha=.8, color="black",
21              label='support vector')
22 plt.legend(loc='best', shadow=False, scatterpoints=1)
23 plt.title('LDA of Wine dataset')
24 plt.grid()
25 plt.show()
26
27 xmin, xmax = [np.min(Z[:,0]), np.max(Z[:,0])]
28 ymin, ymax = [np.min(Z[:,1]), np.max(Z[:,1])]
29 LD = np.linspace(xmin, xmax, 100)
30 LD1 = (-clf.coef_[0,0]*LD - clf.intercept_)/clf.coef_[0,1]
31 LD2 = (-clf.coef_[0,0]*LD +1- clf.intercept_)/clf.coef_[0,1]

```

```

32 LD3 = (-clf.coef_[0,0]*LD -1 - clf.intercept_)/clf.coef_[0,1]
33
34 for color, i, target_name in zip(colors, [0, 1], target_names):
35     plt.scatter(Z[Y == i, 0], Z[Y == i, 1], alpha=.8,
36                 label=target_name)
37 plt.xlabel('LDA1')
38 plt.ylabel('LDA2')
39 plt.scatter(Z[clf.support_, 0], Z[clf.support_, 1], alpha=.8, color="black",
40             label='support vector')
41 plt.legend(loc=1, shadow=False, scatterpoints=1)
42 plt.plot(LD, LD1, color = "red")
43 plt.plot(LD, LD2, color = "red", linestyle = "dashed")
44 plt.plot(LD, LD3, color = "red", linestyle = "dashed")
45 plt.grid()
46 plt.show()

```

ソースコード 3: 問2の主成分分析

```

1 crab = pd.read_csv("crabs.csv", index_col=0)
2
3 # 標準化
4 crabs = crab.iloc[:, 3: ].apply(lambda x: (x-x.mean())/x.std(), axis=0)
5 crabs.head()
6
7 # 主成分分析の実行
8 from sklearn.decomposition import PCA
9 pca = PCA()
10 pca.fit(crabs)
11
12 # データを主成分空間に写像
13 feature = pca.transform(crabs)
14 print(feature.shape)
15
16 Y = crab["sex"].map({'M': 0, 'F': 1})
17
18 # 寄与率
19 pd.DataFrame(pca.explained_variance_ratio_, index=["PC{}".format(x + 1) for
    x in range(len(crabs.columns))])
20
21 # 累積寄与率を図示する
22 import matplotlib.ticker as ticker
23 plt.gca().get_xaxis().set_major_locator(ticker.MaxNLocator(integer=True))
24 plt.plot([0] + list(np.cumsum(pca.explained_variance_ratio_)), "-o")
25 plt.xlabel("Number of principal components")
26 plt.ylabel("Cumulative contribution rate")
27 plt.grid()
28 plt.show()
29
30 # 2次元に射影したデータのプロット
31 import matplotlib.pyplot as plt
32 plt.figure(figsize=(6,6))
33 colors = ['blue', 'red']
34 classes = ['Male', 'Female']
35 for color, i, target_name in zip(colors, [0, 1], Y):

```

```

36     scatter = plt.scatter(feature[Y == i, 0], feature[Y == i, 1], label=
37         target_name)
38 plt.xlabel("PC1")
39 plt.ylabel("PC2")
40 plt.grid()
41 plt.show()
42
43 # デンドログラム
44 from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
45
46 sps = crab.sp.values.tolist()
47 sexes = crab.sex.values.tolist()
48 crab_labels = []
49
50 for sp, sex in zip(sps, sexes):
51     crab_labels.append(sp + sex)
52
53 # 階層型クラスタリングの実施
54 linkage_result = linkage(crab.iloc[:, 3:], method = 'single', metric='
55 euclidean')
56 threshold = 0.4 * np.max(linkage_result[:, 2])
57
58 # 階層型クラスタリングの可視化
59 plt.figure(num=None, figsize=(16, 9), dpi=200, facecolor='w', edgecolor='k
60 ')
59 dendrogram(linkage_result, color_threshold=threshold, labels=crab_labels)
60 plt.show()

```

ソースコード 4: 問3の前処理

```

1 train = pd.read_csv("train.csv")
2 test = pd.read_csv("test.csv")
3
4 # 欠損確認
5 def kesson_table(df):
6     null_val = df.isnull().sum()
7     percent = 100 * df.isnull().sum()/len(df)
8     kesson_table = pd.concat([null_val, percent], axis=1)
9     kesson_table.columns = kesson_table.rename(
10         columns = {0 : '欠損数', 1 : '%'})
11     return kesson_table.columns
12
13 train["Age"] = train["Age"].fillna(train["Age"].median())
14 train["Embarked"] = train["Embarked"].fillna("S")
15
16 train["Sex"][train["Sex"] == "male"] = 0
17 train["Sex"][train["Sex"] == "female"] = 1
18 train["Embarked"][train["Embarked"] == "S"] = 0
19 train["Embarked"][train["Embarked"] == "C"] = 1
20 train["Embarked"][train["Embarked"] == "Q"] = 2
21
22 test["Age"] = test["Age"].fillna(test["Age"].median())
23 test["Sex"][test["Sex"] == "male"] = 0

```

```

24 test["Sex"][test["Sex"] == "female"] = 1
25 test["Embarked"][test["Embarked"] == "S"] = 0
26 test["Embarked"][test["Embarked"] == "C"] = 1
27 test["Embarked"][test["Embarked"] == "Q"] = 2
28 test.Fare[152] = test.Fare.median()
29
30
31 features = train[["Pclass", "Age", "Sex", "Fare", "SibSp", "Parch", "Embarked"]].values
32 features_te = test[["Pclass", "Age", "Sex", "Fare", "SibSp", "Parch", "Embarked"]].values
33 Y_tr = train['Survived'].values
34 Y_te = Y_tr[0:418]

```

ソースコード 5: 問 3 の交差検証法による誤り率と正確度のプロット（決定木）

```

1 from sklearn.model_selection import cross_val_score
2
3 def run_cross_validation_on_trees(X, y, tree_depths, cv=10, scoring='accuracy'):
4     cv_scores_list = []
5     cv_scores_std = []
6     cv_scores_mean = []
7     accuracy_scores = []
8     cv_error = []
9     for depth in tree_depths:
10         #tree_model = DecisionTreeClassifier(max_depth=depth)
11         tree_model = tree.DecisionTreeClassifier(max_depth=depth)
12         cv_scores = cross_val_score(tree_model, X, y, cv=cv, scoring=scoring)
13         cv_scores_list.append(cv_scores)
14         cv_scores_mean.append(cv_scores.mean())
15         cv_scores_std.append(cv_scores.std())
16         accuracy_scores.append(tree_model.fit(X, y).score(X, y))
17         cv_error.append(1-np.mean(cv_scores))
18     cv_scores_mean = np.array(cv_scores_mean)
19     cv_scores_std = np.array(cv_scores_std)
20     accuracy_scores = np.array(accuracy_scores)
21     return cv_scores_mean, cv_scores_std, accuracy_scores, cv_error
22
23 def plot_cross_validation_on_trees(depths, cv_scores_mean, cv_scores_std,
24                                   accuracy_scores, title):
25     fig, ax = plt.subplots(1,1, figsize=(15,5))
26     ax.plot(depths, cv_scores_mean, '-o', label='mean cross-validation
27             accuracy', alpha=0.9)
28     ax.fill_between(depths, cv_scores_mean-2*cv_scores_std, cv_scores_mean+2*
29                     cv_scores_std, alpha=0.2)
30     ax.plot(depths, accuracy_scores, '-*', label='train accuracy', alpha
31             =0.9)
32     ax.set_title(title, fontsize=16)
33     ax.set_xlabel('Tree depth', fontsize=14)
34     ax.set_ylabel('Accuracy', fontsize=14)
35     ax.set_xticks(depths)
36     ax.legend()
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
617
618
619
619
620
621
622
623
624
625
625
626
627
627
628
629
629
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1
```

```

34 sm_tree_depths = range(1,11)
35 sm_cv_scores_mean, sm_cv_scores_std, sm_accuracy_scores, cv_error =
    run_cross_validation_on_trees(features, Y_tr, sm_tree_depths)
36 print(min(cv_error))
37
38 sm_cv_scores_mean, sm_cv_scores_std, sm_accuracy_scores, cv_error =
    run_cross_validation_on_trees(features_te, Y_te, sm_tree_depths)
39 print(min(cv_error))
40
41 import matplotlib.pyplot as plt
42 plot_cross_validation_on_trees(sm_tree_depths, sm_cv_scores_mean,
    sm_cv_scores_std, sm_accuracy_scores, ' ')

```

ソースコード 6: 問3の重要変数

```

1 # ランダムフォレスト
2 clf= RandomForestClassifier(max_depth=4, random_state=0)
3 clf.fit(features, Y_tr)
4 VIP = clf.feature_importances_
5 print('Feature Importances:')
6 names = ["Pclass","Age","Sex","Fare", "SibSp", "Parch", "Embarked"]
7 for i, feat in enumerate(names):
8     print('\t{0:20s} : {1:>.6f}'.format(feat, VIP[i]))
9
10 import matplotlib.pyplot as plt
11 indices = np.argsort(VIP)
12 plt.figure()
13 plt.title("Feature importances")
14 plt.barh(range(features.shape[1]), VIP[indices], color="r", align="center")
15 names = ["Embarked","Parch","SibSp","Age", "Fare", "Pclass", "Sex"]
16 plt.yticks(range(features.shape[1]), names)
17 plt.ylim([-1, features.shape[1]])
18 plt.show()
19
20 # アダブースト
21 clf = AdaBoostClassifier(n_estimators=200, random_state=2020)
22 clf.fit(features, Y_tr)
23 VIP = clf.feature_importances_
24 print('Feature Importances:')
25 names = ["Pclass","Age","Sex","Fare", "SibSp", "Parch", "Embarked"]
26 for i, feat in enumerate(names):
27     print('\t{0:20s} : {1:>.6f}'.format(feat, VIP[i]))
28
29 import matplotlib.pyplot as plt
30 indices = np.argsort(VIP)
31 plt.figure()
32 plt.title("Feature importances")
33 plt.barh(range(features.shape[1]), VIP[indices], color="r", align="center")
34 names = ["Embarked","Parch","Pclass","Sex", "SibSp", "Age", "Fare"]
35 plt.yticks(range(features.shape[1]), names)
36 plt.ylim([-1, features.shape[1]])
37 plt.show()

```
