Python と機械学習

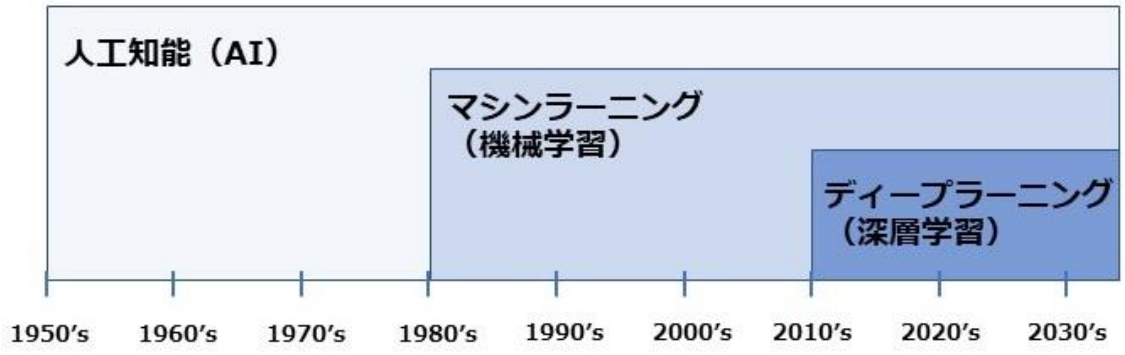
１．はじめに

Python チュートリアルに参加する機会がありました。

機械学習について、sklearn というライブラリーの決定木というアルゴリズムを学んだので、

データで取り組むとともに、内容を解説します。

1.1 機械学習 について



「機械学習」は、上図に示すように、1950 年代から提唱された「人工知能」の一つとなります。

データから規則性や判断基準を学習し、未知のものを予測、判断します。

さらに、2010 年代から「深層学習」が出てきますが、これは、機械学習のうちニューラルネットワークと呼ばれる手法の進化形です。

（深層学習は、本編では登場しません。可能であれば、次回説明します。）

「機械学習」は、三つに分類されます。

・教師あり学習 --- 分析対象と教師データとがあり、教師データを参考に、回帰（予測）
分類（識別）を行います。

・教師なし学習 --- 分析対象のみから学習します。データの共通項、法則性を見つけ出して
クラスタリング、次元削除を行います。

・強化学習 --- 試行錯誤の中で、最適方策を調整します。
ゲーム、ロボットの歩行学習が代表例です。

今回は、Python で、教師あり学習の決定木モデルを作成します。

使用するライブラリは、 sklearn, pandas, seaborn です。

1.2　使用したデータ

　決定木モデルの解説には、タイタニック号沈没データを使う事が多いようですが、

今回は、新型コロナの感染状況についてのデータで、行います。

　大阪府の新型コロナの感染状況

・期間　　2020 年　1 月 29 日　から　5 月 1 6 日

・データ数　1770　（参考　死亡者は　72 人でした）

このデータは、当時、大阪府から毎日公開されていた、日毎の感染状況をまとめたものです。

（当時、4 月から在宅勤務になり、Python の勉強をしてました。

　pandas　を　新型コロナ感染状況のデータ整理で勉強していたので、その時収集したデータです）

大阪府の公開内容の一例です。　（様式は何回か変更されました。以下が最終様式）

番号	年代	性別	居住地		同居家族	職業	発症日	症状	濃厚接触者	特記事項
388	20	男性	堺	市	調査中	自営業	3/27	軽症		
389	30	男性	堺	市	あり	会社員	3/26	軽症		基礎疾患あり 発症後はマスク着用

上記の内容で、新規陽性者が、毎日、発表されてました。

また、死亡者が出ると、番号　と　死亡日　が　発表されてました。

第一波ということもあり、また、新規陽性者数も 1 日の最大が 97 名でしたので、詳しい内容で、データをまとめられたのではと推察します。（2022 年 11 月の今は 1 日で 3000 名をオーバーしています）

現在も、以下の URL で、発生状況を確認できます。

大阪府／新型コロナウイルス感染症患者の発生状況（令和 2 年 11 月 2 日以降）（osaka.lg.jp）

２．プログラムの説明と解析結果

2.1 CSV データを作成する

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	age	gender	onset	onset(s)	found	found(S)	day death	day death	death	symptom	conatact	location	ud
2	40	1	2020/1/20	43850	2020/1/29	43859			0	1	1	fu	0
3	40	0	2020/2/20	43881	2020/2/27	43888			0	1	1	city	0
4	40	1	2020/2/19	43880	2020/2/28	43889			0	1	1	fu	0
5	5	1	2020/2/27	43888	2020/2/27	43888			0	1	1	fu	0

上記の CSV データを作成します。

age --- 年代で公開されてるので、10 刻みになります。

gender --- 0 が 男性 、1 が 女性です。

onset --- 発症日 （日付表示です）

onset(s) --- 発症日のシリアル表示です。

found --- 陽性判別日

found(S) --- 陽性判別日のシリアル表示です。

day death --- 死亡日

day death(S) --- 死亡日のシリアル表示です。

death --- 0 が 生存 、1 が 死亡です。

symtom --- 症状 0 が 無症状、 1 が 軽症 、 2 が 重症です。

contact --- 0が 濃厚接触なし 1が濃厚接触あり です。

location --- 居住地を入力してます。

ud --- 0 が 基礎疾患なし、 1 が 基礎疾患あり です。

＊シリアル値とは、「1900 年 1 月 1 日」を「1」として、何日経過したかを示す数値です。

「日付」表示を、「標準」、「数値」 の表示形式にすると、シリアル値の表示になります。

CSV の入力は、府からの発表データを、カット＆ペーストしてました。

日々のデータ(新規陽性者数)は最高の日で 90 個程度でしたので、出来たと思います。

2.2 ライブラリの説明

・pandas　データ分析のためのライブラリで、データの読み込み、加工、分析を容易に行います。

機械学習において、データの前処理のため用いられます。

データ構造として、1 次元の Series、2 次元の DataFrame　があります。

・seaborn　データを可視化するライブラリで、matplotlib　が一般ですが、見やすい可視化を

実現します。heatmap 関数で、相関係数を色別で可視化します。

・sklearn.preprocessing import LabelEncoder

文字列で表されたラベルを、0 ～(ラベル種類数-1)　までの数値に変換します。

・sklearn.model_selection import train_test_split

データを訓練用とテスト用に分割します。

test_size :テスト用の割合指定

random_state=0 :乱数シードを固定する

stratify :　均等に分割させたいデータを指定して、そのデータの比率を一致させる。

・sklearn.tree import DecisionTreeClassfier

criterion=　デフォルトは、gini,　 entropy, log_loss　が選択可

splitter=デフォルトは　best,　　random　が選択可

max_depth=　ツリーの最大深さ　数字で指定

・sklearn.tree import import plot_tree

決定木を可視化する

2.3 IDE(統合開発環境)について、

IDE とは、プログラムを入力する環境です。入力して、デバグして、保存という順番になります。

IDE　は、2　種類あって、対話形式と、プログラムを入力する所とが結果を出力する所が異なるビューを持つ形式があります。今回は、後者の形式で検討してるので、報告書の中での記載は、出力内容をプログラムの中に、挿入する体裁としています。実際は、入力と出力は別のビューです。

対話形式の IDE として、Jupyter Motebook　、対話型でない IDE として　VSCode, PyCharm がよく使われているようです。

（ちなみに、今回使用の IDE は、初心者向けの　Thonny　という IDE です）

2.4 プログラムの説明

----------プログラム----------

```
#基本ライブラリ
import pandas as pd
import pprint
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv("corona_20200129to0516.csv")
print(type(df))
print(df)
print(df.columns)
```

＃の後は、コメント文でプログラムではないです

import ○○○ でライブラリをインポートします。
numpy, pandas ,matplotlib は標準的なものです。

corona_20200129to0516.csv というデータを読み込んでいます。フォルダが異なれば、path の指定が必要です。

データの内容を確認します。type は データの型の出力させて、
print(df) は省略されますが、全データの表示です。
columns は、列の名称を出力します。

pandas で作成したデータの出力です
13 列 1770 行 と表示されてます。

```
<class 'pandas.core.frame.DataFrame'>
      age   gender      onset  onset(s)  ... symptom  conatact location  ud
0      40        1  2020/1/20     43850  ...       1         1       fu   0
1      40        0  2020/2/20     43881  ...       1         1     city   0
2      40        1  2020/2/19     43880  ...       1         1       fu   0
3       5        1  2020/2/27     43888  ...       1         1       fu   0
4      50        0  2020/2/22     43883  ...       1         0       fu   0
...   ...      ...        ...       ...  ...     ...       ...      ...  ..
1765   20        1  2020/5/13     43964  ...       1         0     city   0
1766   60        1   2020/5/2     43953  ...       1         1       fu   0
1767   80        1  2020/5/13     43964  ...       1         0       fu   0
1768   70        1  2020/5/13     43964  ...       1         1       fu   0
1769   80        1  2020/5/13     43964  ...       1         1       fu   0

[1770 rows x 13 columns]
Index(['age ', 'gender', 'onset', 'onset(s)', 'found', 'found(s)', 'day death',
       'day death(s)', 'death', 'symptom', 'conatact', 'location', 'ud'],
      dtype='object')
```

column 名を、出力しました。
‘age ‘ とあります。誤って後ろにスペースを入力しました。
‘age’ で入力すると、エラーになります。（当たり前ですが）

```
df["location"].fillna(df["location"].mode()[0],inplace=True)
print(df["location"].hasnans)
print(df.iloc[1350:1380,11])
```

fillna は欠損値処理を行います。
Mode() で 最頻値に置き換えます。
hasnans は、欠損値に有無を確認します。無は False
最後に処理できたか、出力確認してます。

```
False
1350    city
1351    city
          （1352 から 1377 までは省略）
1378    city
1379    city
Name: location, dtype: object
```

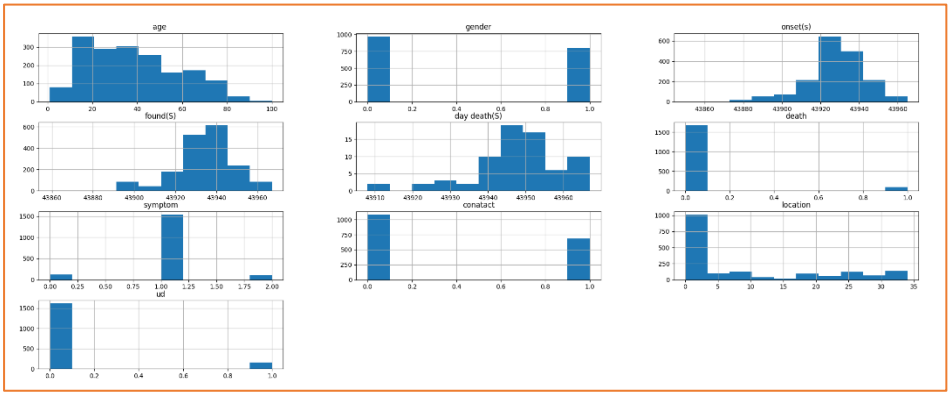
city （大阪市のこと)が
最頻値だったので、全て
city に変換されました。

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df["location"] = le.fit_transform(df["location"])
print(df["location"])
```

Location 列のデータは、文字列ですので、数値に変換します。
（カテゴリカル変数を数値変数に変換 難しく言うと）
33 の都市があったので、0～33 の数字に変換されました

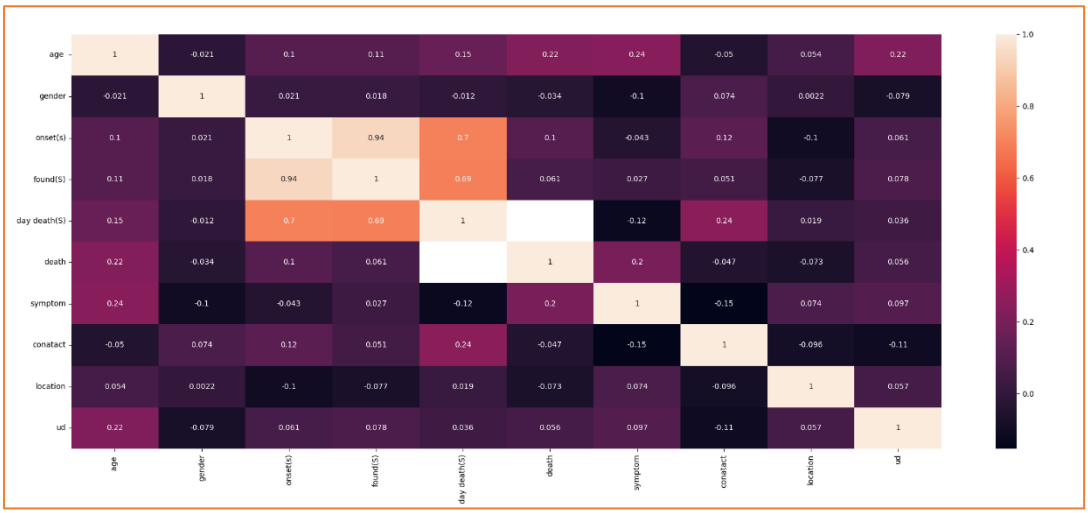
```
df.hist(figsize=(12,12))
plt.show()
```

変数毎のヒストグラムを作成しました。
拡大したものが必要なら、連絡ください。



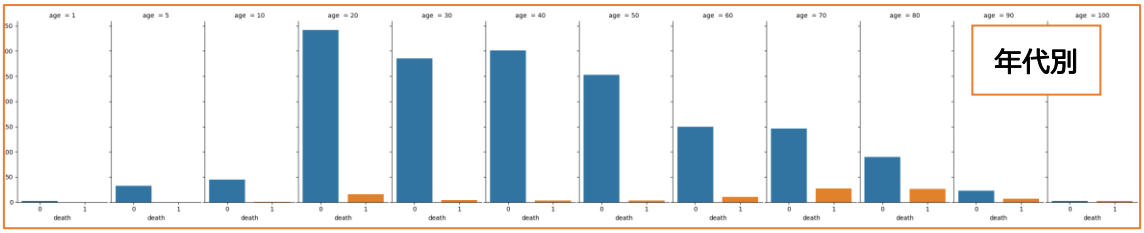
変数毎のヒートマップを出力しました。
拡大したものは、報告書の最後に載せます。

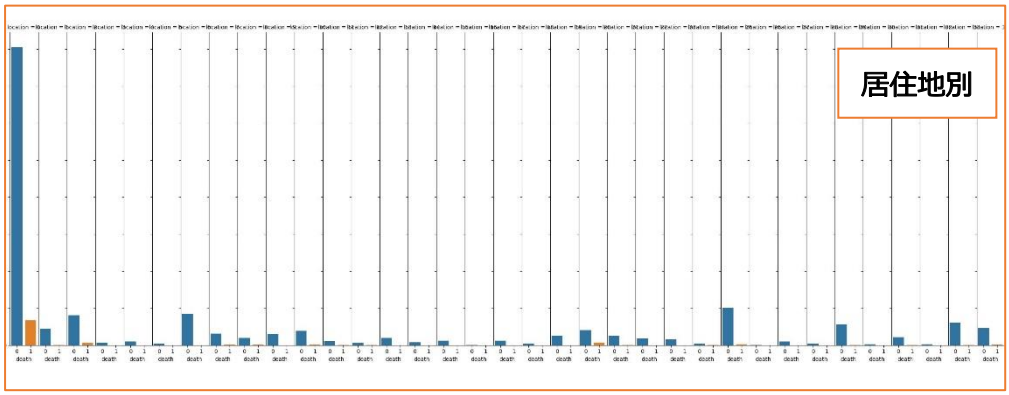
```
import seaborn as sns
_, ax = plt.subplots()
sns.heatmap(df.corr(),annot=True, ax=ax)
plt.show()
```

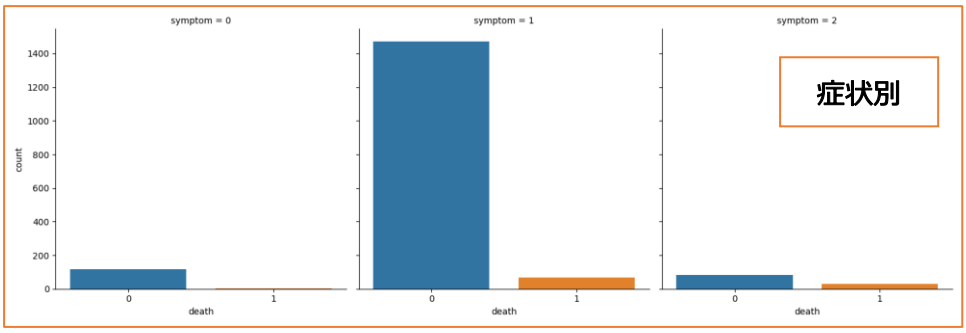


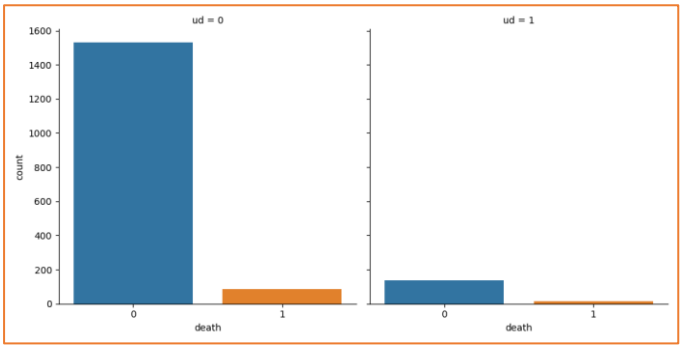
死亡について、年代、居住地、症状、基礎疾患　との関連を出力しました。
拡大したものが必要なら、連絡ください。

```
print(sns.catplot(x="death", col="age ", kind="count",data=df))
print(sns.catplot(x="death", col="location", kind="count",data=df))
print(sns.catplot(x="death", col="symptom", kind="count",data=df))
print(sns.catplot(x="death", col="ud", kind="count",data=df))
```









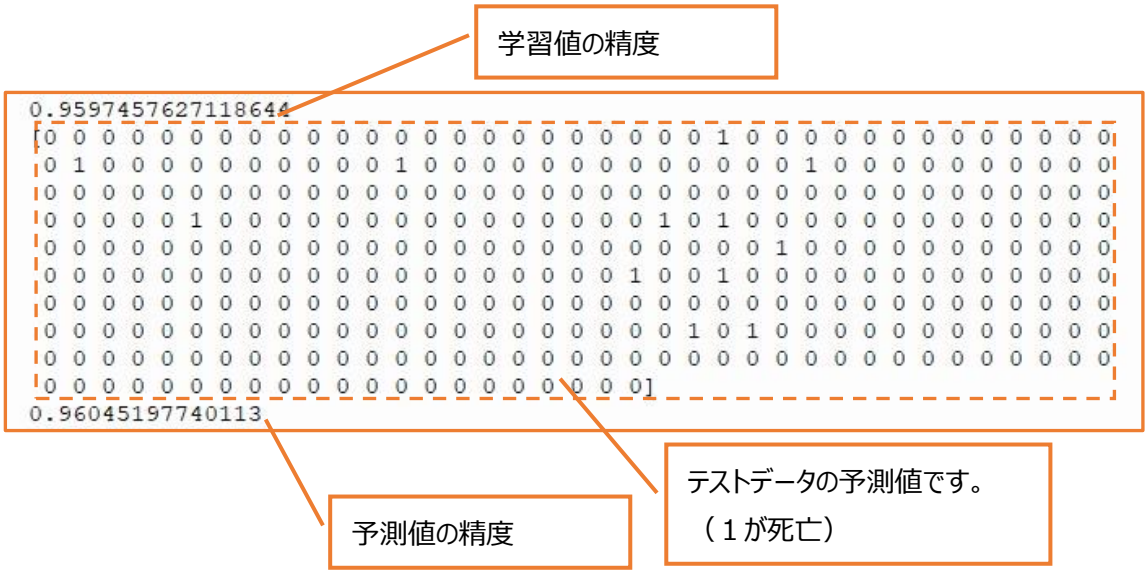
基礎疾患別

学習データ(_train)とテストデータ(_test)に分けます。
X が説明変数、y が正解ラベル（ここでは死亡）とします。
X について、“onset”,”found”,”day death”は日付表示で、シリアル表示があるので、除外します。死亡を示す “day death(S)”, “death” も除外しました。
y は “death” のみ（0 か 1）のデータです。
テストデータ を 20%にするよう区分けしました。

```
from sklearn.model_selection import train_test_split
X = df.drop(["onset","found","day death","day death(S)","death"], axis=1)
y = df["death"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ¥
random_state=0, stratify=y)
```

決定木モデルを作成します。
model.score で 学習データの精度を出力します。
model.predict で、テストデータの予測を出力し、
accuracy_score でテストデータの精度を算出します。

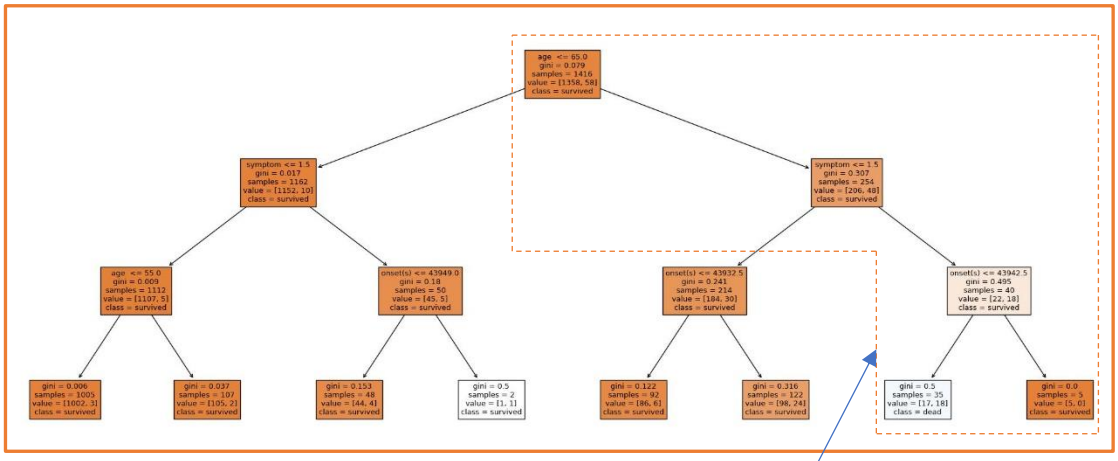
```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(max_depth=3)
model.fit(X_train, y_train)
print(model.score(X_train, y_train))
pred = model.predict(X_test)
print(pred)
from sklearn.metrics import accuracy_score
print(accuracy_score(pred, y_test))
```



決定木モデルを可視化します。

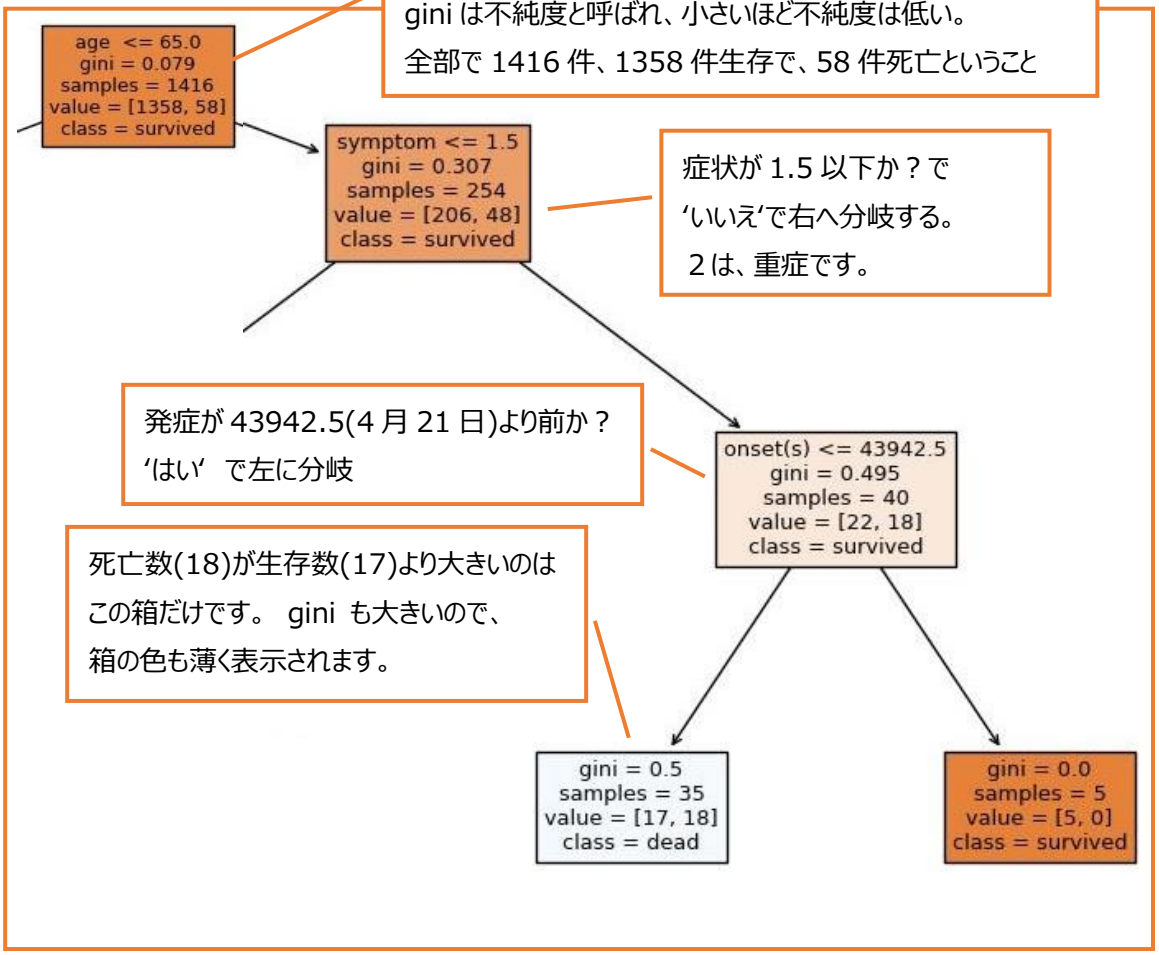
```
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree
plt.figure(figsize=(15, 10))
plot_tree(model,feature_names=X_test.columns,class_names=["survived",¥
“dead"], filled=True)
plt.show()
```

決定木モデルを可視化したものです
拡大したものは、報告書の最後に載せますが、部分的に拡大して、一部説明します。



点線部を拡大して、説明します。

65 才以下か？で ‘いいえ’で右へ分岐する。
gini は不純度と呼ばれ、小さいほど不純度は低い。
全部で 1416 件、1358 件生存で、58 件死亡ということ



症状が 1.5 以下か？で
‘いいえ’で右へ分岐する。
2 は、重症です。

発症が 43942.5(4 月 21 日)より前か？
‘はい‘ で左に分岐

死亡数(18)が生存数(17)より大きいのは
この箱だけです。 gini も大きいので、
箱の色も薄く表示されます。

今回は、決定木モデルを可視化したところで、機械学習の項は、終わりです。

決定木を見ての結果ですが、

・「60 代以下で、症状が軽症、無症状の方は、死亡の例がすくない」と言えます。

学習データ 1416 件の中で、1112 件が該当し、内、死亡件数は、5 例でした。

gine 係数は、 0.009 でした。

更に「50 代以下で、症状が軽症、無症状の方は、死亡の例が、更にすくない」と言えます。

学習データ 1416 件の中で、1005 件が該当し、内、死亡件数は、3 件でした。

gine 係数は、 0.006 でした。

・「70 代以上で、症状が重症の方は、死亡件数が、生存件数を上回りました」

学習データ 1416 件の中で、35 件が該当し、内、死亡件数は、18 件でした。

gine 係数は、 0.5 で、 不純度は高くなってます。

また、このケースには、「4 月 21 日前に発症した」という条件が付きますが、

今回採取したデータは、5 月 16 日までです。

5 月 16 日までに、陽性になり、5 月 16 日以降に死亡された方は、カウントされてません。

故に、発症日が 5 月 16 日に近い事例では、死亡されてもデータ化されてない可能性はあります。

（4 月 21 日以前であれば、死亡が 5 月 16 日までにカウントされやすいということ）

最後に追記ですが、このモデルでは、不十分ということで、決定木モデルを複数、用意する、

「ランダムフォレスト」という手法があるようです。

機会あれば、勉強してみます。

2.5 最後に、大きくしたグラフを一部、追加します。

2.5.1 ヒートマップ

縦軸、横軸に

変数を置いてます。

数字は相関係数です。

（見にくいですが）

相関係数によって、

マスの色合いを

変えています。

発症日、陽性判定日、

死亡日の相関が

高いですが、

これは、上記の順番で

事象は発生しますし、

間隔も、一定に

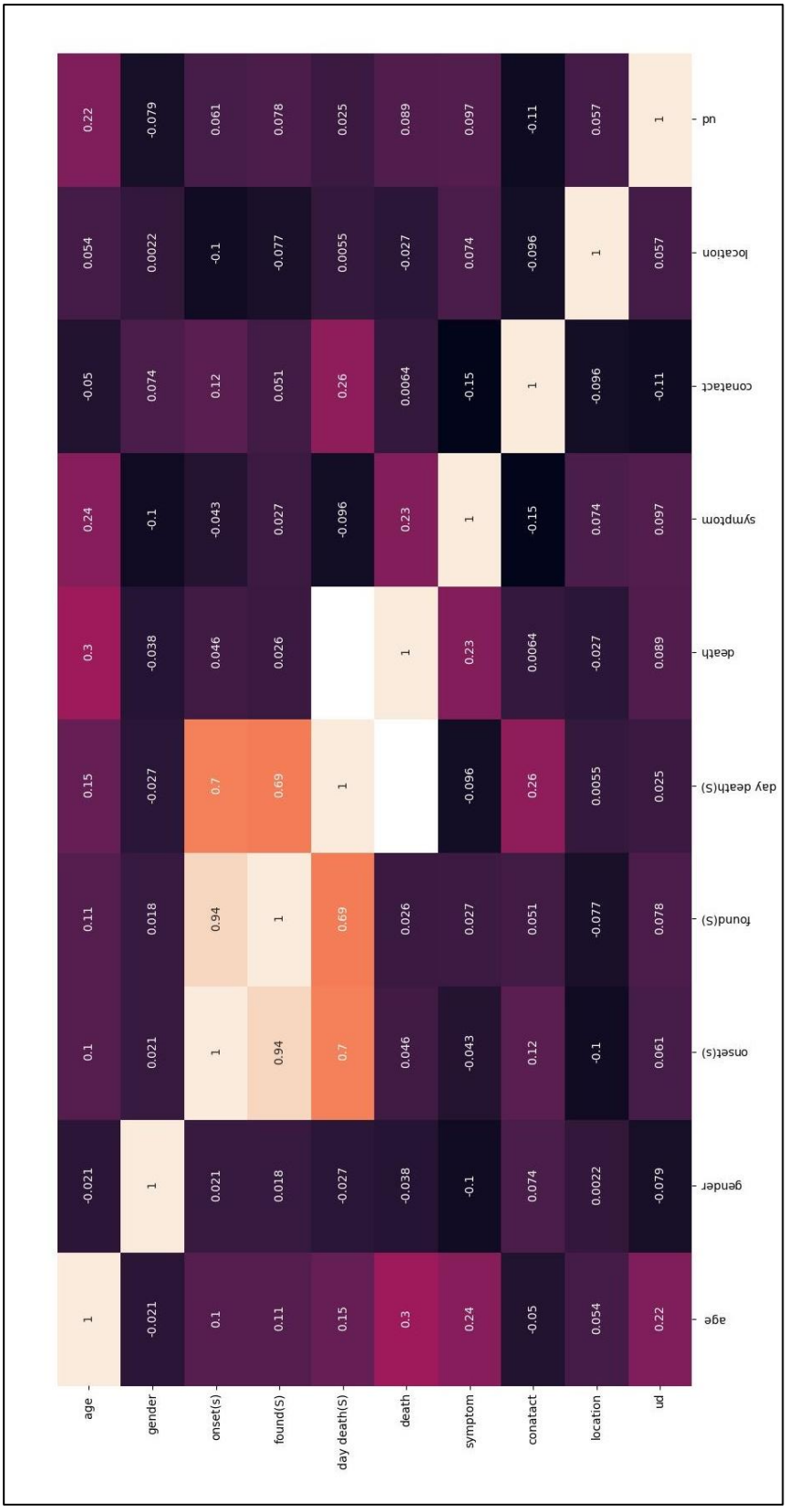
近かったと考えます。

年代と死亡の相関は、

0.3 でした。

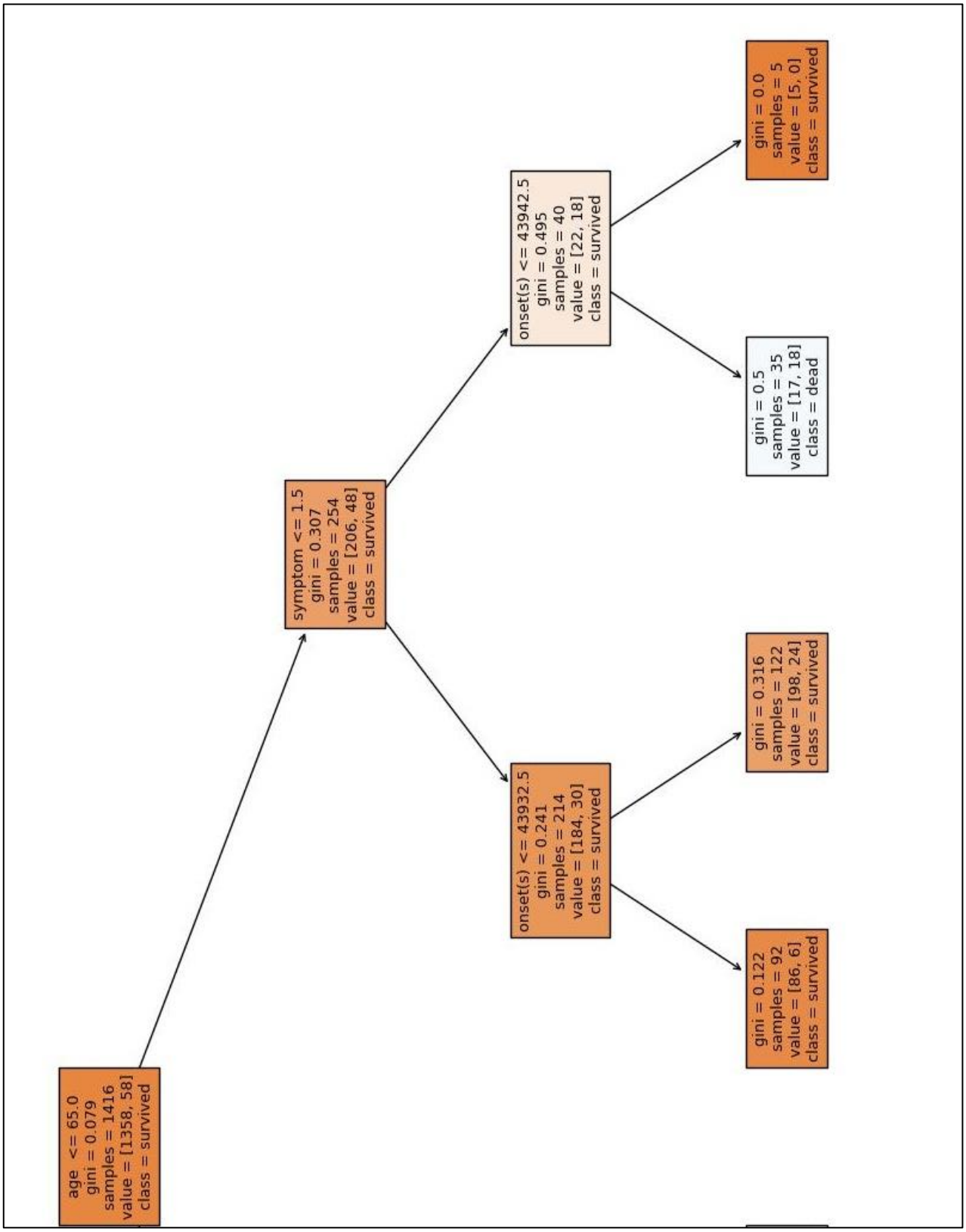
「弱い相関」と

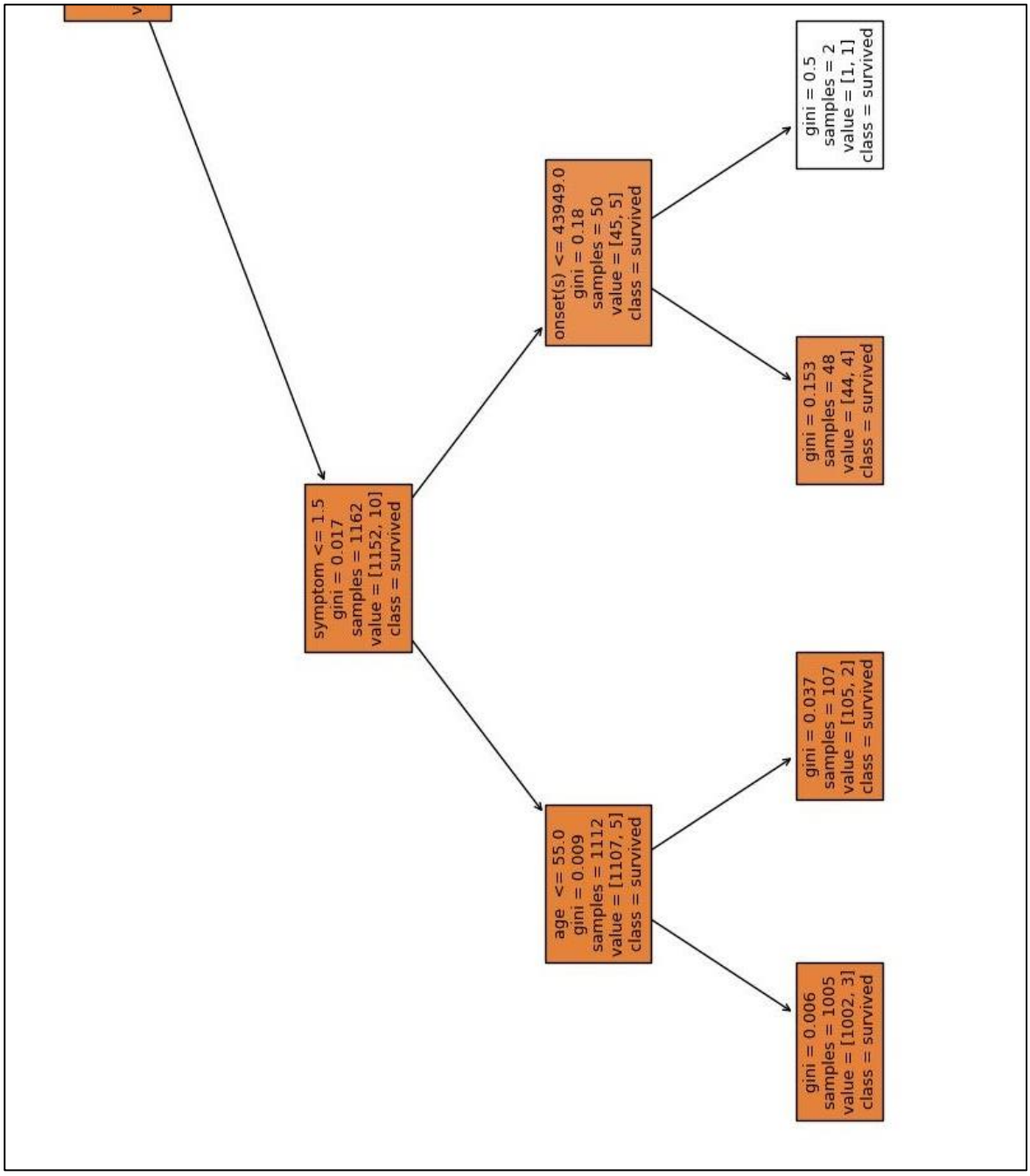
言えます。



2.5.2 決定木

説明は前項で行いました。 縦に 2 分割して、貼付けます。




symptom <= 1.5
gini = 0.017
samples = 1162
value = [1152, 10]
class = survived
onset(s) <= 43949.0
gini = 0.18
samples = 50
value = [45, 5]
class = survived
gini = 0.5
samples = 2
value = [1, 1]
class = survived
gini = 0.153
samples = 48
value = [44, 4]
class = survived
age <= 55.0
gini = 0.009
samples = 1112
value = [1107, 5]
class = survived
gini = 0.037
samples = 107
value = [105, 2]
class = survived
gini = 0.006
samples = 1005
value = [1002, 3]
class = survived