**Pythonを利用したAMLの悪性形質の規定因子の特定**

**導入（解決すべき社会課題）**

がんを根治できる夢のような抗がん剤を開発するために、私はがんの実態解明と新たな治療薬の開発研究に従事してきました。一部のがんでは抗がん剤に完治できるようにきたものの、多くのがんは完治につながる有効な治療薬はない。白血病の一種である急性骨髄性白血病(AML)は予後不良の悪性腫瘍の一種で、その発祥の原因遺伝子が多数特定され、新たながん治療薬の開発が進められてきました。それでもなお、抗がん剤で完治できるAMLは1部のサブタイプに限定されいます。AMLのより有効性の高い治療薬を開発するためには、AMLの臨床データーに基づき、悪性形質の原因遺伝子を新たに特定し、新たな治療標的を特定することが重要であると思われます。今回は2018年にnature誌で報告されたAML患者サンプルのデーター[1]を用いて、AMLの悪性形質の規定の探索を行いました。また、悪性形質の規定因子の発現量のデーターは生存期間（AMLの悪性度）の予測に利用しうるかも検討しました。

**分析するデーター**

2018年にnature誌で報告されたAML患者サンプルの遺伝子発現データーおよび生存期間のデーターのみを使用して、AMLの悪性形質の規定分子を探索しました。

**実行環境**

パソコン：FUJITSU (Windows 11)

開発環境：Google Corroboratory

言語：Python

ライブラリー：Panda, Matplotlib等

**分析の流れ**

1. 前処理（行と列の転置、カラムを数値型に変換）
2. 生存期間と相関性の高い遺伝子を特定
3. 回帰で生存期間を予測（悪性度の予想）

**分析の過程**

**import numpy as np**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**data = pd.read\_csv('patient\_data.csv')**

**#dataチェック**

**data**

**#データー形を確認**

**data.info()**

**##########################################前処理##############################**

**# 転置**

**data = data.T**

**# 転置の確認**

**data**

**# 欠損値の確認**

**data.isnull().values.any()**

**# インデックスをリセットしてから1行目をカラム名に設定**

**data = data.reset\_index(drop=True)**

**# 1行目をカラム名に設定**

**data.columns = data.iloc[0]**

**# 1行目を削除してインデックスをリセット**

**data = data.drop(0).reset\_index(drop=True)**

**# データーの確認**

**data**

**# データーの型を確認**

**data.info()**

**# すべてのカラムを数値型に変換 (エラーがあれば NaN に置換)**

**data = data.apply(pd.to\_numeric, errors='coerce')**

**# 変換後のデータ型を確認**

**data.info()**

**# データーの確認**

**data**

**#######################################Surviveとの相関する遺伝子を抽出##############################**

**#数値データの相関行列を作成**

**correlation\_matrix = data.corr()**

**# Surviveとの相関する遺伝子を抽出（デフォルトはピアソン係数）**

**Survive\_correlation = correlation\_matrix['Survive'].sort\_values(ascending=False)**

**# 0.2以上または-0.2以下の要素を抽出**

**high\_correlation\_features= Survive\_correlation[(Survive\_correlation >= 0.2) | (Survive\_correlation <= -0.2)]**

**print(high\_correlation\_features)**

**# 高相関特徴量とsurvivalのデータを抽出**

**selected\_features = high\_correlation\_features.index.tolist()**

**# 特徴量名を取得するためのマッピングを作成**

**encoded\_columns = data.columns.tolist()**

**# マッピングを作成**

**mapping = {col: col for col in encoded\_columns}**

**# 特徴量名を取得**

**selected\_features = [mapping.get(col, col) for col in selected\_features]**

**# 元のtrainデータフレームから特徴量とsurvivalを抽出**

**data\_selected = data[selected\_features]**

**print(data\_selected.head())**

**#カラムの数を確認**

**num\_columns = data\_selected.shape[1]**

**print("カラムの数:", num\_columns)**

**######################################相関性のグラフ化##########################################**

**# カラム数**

**features = data\_selected.columns**

**# 図を4列に並べる設定**

**n\_cols = 4**

**n\_rows = int(np.ceil(len(features) / n\_cols)) # 行数を計算**

**fig, axes = plt.subplots(n\_rows, n\_cols, figsize=(20, n\_rows \* 5))**

**axes = axes.flatten() # 1次元配列に変換**

**# 各特徴量について散布図を描画**

**for i, feature in enumerate(features):**

**# 回帰直線と相関係数を表示**

**sns.regplot(data=data\_selected, x=feature, y='Survive', ax=axes[i], scatter\_kws={'s':10}, line\_kws={'color':'red'})**

**# 相関係数の計算**

**correlation = data\_selected[feature].corr(data\_selected['Survive'])**

**# タイトルに相関係数を追加**

**axes[i].set\_title(f'Survive vs {feature}\nCorrelation: {correlation:.2f}')**

**axes[i].set\_xlabel(feature)**

**axes[i].set\_ylabel('Survive')**

**# 残った空白のプロットを非表示に**

**for j in range(i + 1, len(axes)):**

**fig.delaxes(axes[j])**

**plt.tight\_layout() # レイアウトを自動調整**

**plt.show()**

**###############################学習##############################################**

**# 説明変数と目的変数の決定**

**train\_features = data.drop('Survive', axis=1).values**

**train\_target = data['Survive'].values**

**from sklearn.model\_selection import train\_test\_split**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(train\_features, train\_target, test\_size=0.3, random\_state=0)**

**# モデルの学習**

**import lightgbm as lgb**

**from sklearn.metrics import mean\_squared\_error, r2\_score**

**# モデルの定義（ハイパーパラメータを手動で設定）**

**model = lgb.LGBMRegressor(**

**n\_estimators=100, # 木の数**

**max\_depth=5, # 木の最大深さ**

**learning\_rate=0.1, # 学習率**

**num\_leaves=31, # 葉の数**

**random\_state=0**

**)**

**# モデルの訓練**

**model.fit(X\_train, y\_train)**

**# テストデータで予測**

**y\_pred = model.predict(X\_test)**

**# MSE（平均二乗誤差）の計算**

**mse = mean\_squared\_error(y\_test, y\_pred)**

**print("テストデータでの平均二乗誤差 (MSE):", mse)**

**# 説明変数と目的変数の決定**

**train\_features = data\_selected.drop('Survive', axis=1).values**

**train\_target = data\_selected['Survive'].values**

**from sklearn.model\_selection import train\_test\_split**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(train\_features, train\_target, test\_size=0.3, random\_state=0)**

**# モデルの学習**

**import lightgbm as lgb**

**from sklearn.metrics import mean\_squared\_error, r2\_score**

**# モデルの定義（ハイパーパラメータを手動で設定）**

**model = lgb.LGBMRegressor(**

**n\_estimators=100, # 木の数**

**max\_depth=5, # 木の最大深さ**

**learning\_rate=0.1, # 学習率**

**num\_leaves=31, # 葉の数**

**random\_state=0**

**)**

**# モデルの訓練**

**model.fit(X\_train, y\_train)**

**# テストデータで予測**

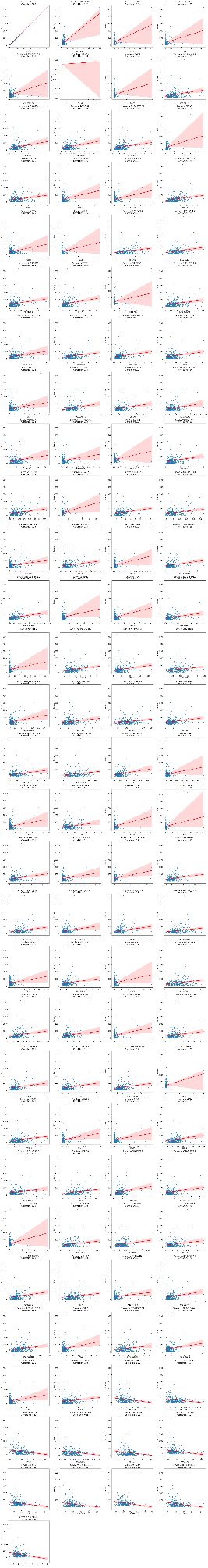
**y\_pred = model.predict(X\_test)**

**# MSE（平均二乗誤差）の計算**

**mse = mean\_squared\_error(y\_test, y\_pred)**

**print("テストデータでの平均二乗誤差 (MSE):", mse)**

**データー**



**分析から得られた情報**

今回の解析により、AMLの悪性形質の原因遺伝子の候補として、116遺伝子を特定しました。特に、EEF1A1P12の発現量が生存期間と強い相関性を示すことが明らかとなりました。また、その遺伝子のみを使用して回帰にて生存期間（悪性度）を推測したところ、すべての遺伝子のデーターで分析した場合よりも、悪性形質の原因遺伝子の候補遺伝子（116遺伝子）にて分析した方が精度高く推測でたものの、その精度は高くはなかった。

**まとめ**

以上の解析から、AMLの新たな悪性形質の規定因子の特定に成功しました。これは新たな治療薬の開発につながる社会的に意義深い発見だと思われます。ただ、精度高く生存期間を予測するには、116遺伝子の発現量を使用して回帰で推定すだけでは不十分で、より詳細な解析が必要であることも明らかとなりました。

**参考文献**

[1] Tyner JW, Tognon CE, Bottomly D, Wilmot B, Kurtz SE, Savage SL, et al. Functional

genomic landscape of acute myeloid leukaemia. Nature. 2018;562:526–31