Nama : Sayyid Abdullah

NPM : 2206130800

UTS Komputasi Lanjut dan Big Data

Opsi 1 Visualisasi Data

Data yang digunakan berupa data cancer. Dalam data tesebut ada sebanyak 30 variabel independen dan 1 variabel dependen yaitu target. Disini akan divisualisasikan variabel targetnya. Skrip dan hasil visualisasinya menggunakan google collab sebagai berikut

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.datasets import load\_breast\_cancer

data = load\_breast\_cancer()

df=pd.DataFrame(data.data, columns=data.feature\_names)

df['target'] = data.target

dftarget=data.target

counts = np.bincount(data.target)

print(counts)

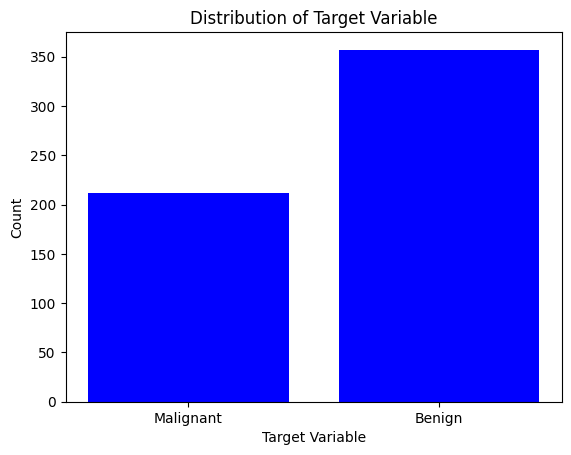
plt.bar(['Malignant', 'Benign'], counts, color='blue')

plt.xlabel('Target Variable')

plt.ylabel('Count')

plt.title('Distribution of Target Variable')

plt.show()



Dari hasil visualisasi data tersebut diperoleh bahwa penderita kanker ganas sebanyak 212 orang dan penderita kanker ringan sebanyak 357 orang.

Dari data tersebut akan dimodelkan menggunakan decision tree, random forest, dan self-training.

Petama menggunakan decision tree diperoleh

from sklearn.datasets import load\_breast\_cancer

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

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data.data, data.target, random\_state=42)

dt = DecisionTreeClassifier()

dt.fit(X\_train, y\_train)

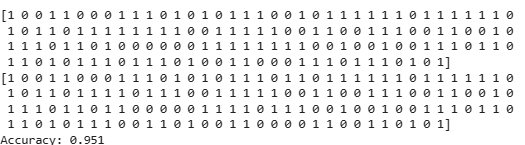
y\_pred = dt.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print(y\_test)

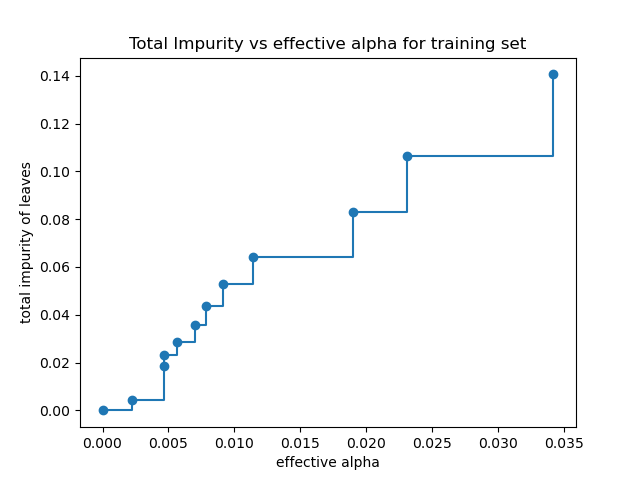
print(y\_pred)

print(f"Accuracy: {accuracy:.3f}")



Hasil yang diperoleh pada bagian array pertama adalah hasil y tesnya dan untuk array kedua adalah y prediksinya dengan akurasi 0,951.

Langkah selanjutnya adalah melihat keterkaitan antara total impurity dengan alfa efektif tanpa memasukkan alfa efektif maksimum karena node trivial. Diperoleh gambar seperti berikut



Selanjutnya adalah train model dengan alfa efektif dan diperoleh nilai alfa yang memangkas seluruh pohon pada simpul terakhir

clfs = []

for ccp\_alpha in ccp\_alphas:

    clf = DecisionTreeClassifier(random\_state=0, ccp\_alpha=ccp\_alpha)

    clf.fit(X\_train, y\_train)

    clfs.append(clf)

print(

    "Number of nodes in the last tree is: {} with ccp\_alpha: {}".format(

        clfs[-1].tree\_.node\_count, ccp\_alphas[-1]

    )

)

Diperoleh output



Langkah selanjutnya adalah melihat keterkaitan antara jumlah node dan kedalaman pohon dengan alfa dengan skrip dan hasil sebagai berikut

clfs = clfs[:-1]

ccp\_alphas = ccp\_alphas[:-1]

node\_counts = [clf.tree\_.node\_count for clf in clfs]

depth = [clf.tree\_.max\_depth for clf in clfs]

fig, ax = plt.subplots(2, 1)

ax[0].plot(ccp\_alphas, node\_counts, marker="o", drawstyle="steps-post")

ax[0].set\_xlabel("alpha")

ax[0].set\_ylabel("number of nodes")

ax[0].set\_title("Number of nodes vs alpha")

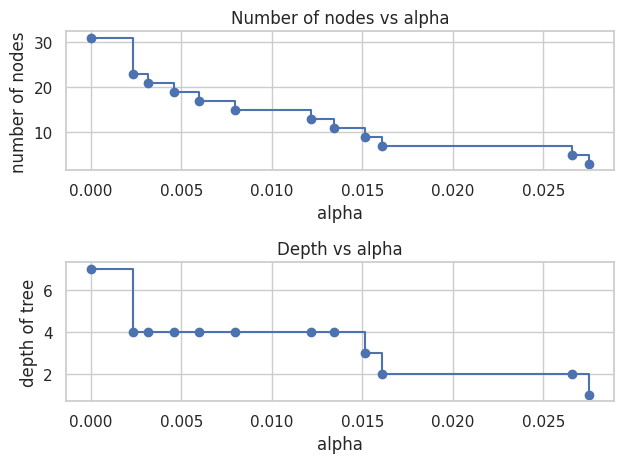
ax[1].plot(ccp\_alphas, depth, marker="o", drawstyle="steps-post")

ax[1].set\_xlabel("alpha")

ax[1].set\_ylabel("depth of tree")

ax[1].set\_title("Depth vs alpha")

fig.tight\_layout()



Semakin besar nilai alfa maka jumlah node dan kedalamannya akan semakin berkurang.

Kedua adalah dengan menggunakan model random forest diperoleh from collections import defaultdict

import matplotlib.pyplot as plt

import numpy as np

from scipy.stats import spearmanr

from scipy.cluster import hierarchy

from scipy.spatial.distance import squareform

from sklearn.datasets import load\_breast\_cancer

from sklearn.ensemble import RandomForestClassifier

from sklearn.inspection import permutation\_importance

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

data = load\_breast\_cancer()

X, y = data.data, data.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=42)

clf = RandomForestClassifier(n\_estimators=100, random\_state=42)

clf.fit(X\_train, y\_train)

print("Accuracy on test data: {:.2f}".format(clf.score(X\_test, y\_test)))



Diperoleh akurasi sebesar 0,97

Selanjutnya memplot kepentingan fitur berbasis pohon dan kepentingan permutasi. Plot kepentingan permutasi menunjukkan bahwa mengubah fitur menurunkan akurasi paling banyak 0,012, yang menunjukkan bahwa tidak ada fitur yang penting. Ini bertentangan dengan akurasi tes yang dihitung di atas: beberapa fitur harus penting. Pentingnya permutasi dihitung pada set train untuk menunjukkan seberapa banyak model bergantung pada setiap fitur selama training.

result = permutation\_importance(clf, X\_train, y\_train, n\_repeats=10, random\_state=42)

perm\_sorted\_idx = result.importances\_mean.argsort()

tree\_importance\_sorted\_idx = np.argsort(clf.feature\_importances\_)

tree\_indices = np.arange(0, len(clf.feature\_importances\_)) + 0.5

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 8))

ax1.barh(tree\_indices, clf.feature\_importances\_[tree\_importance\_sorted\_idx], height=0.7)

ax1.set\_yticks(tree\_indices)

ax1.set\_yticklabels(data.feature\_names[tree\_importance\_sorted\_idx])

ax1.set\_ylim((0, len(clf.feature\_importances\_)))

ax2.boxplot(

    result.importances[perm\_sorted\_idx].T,

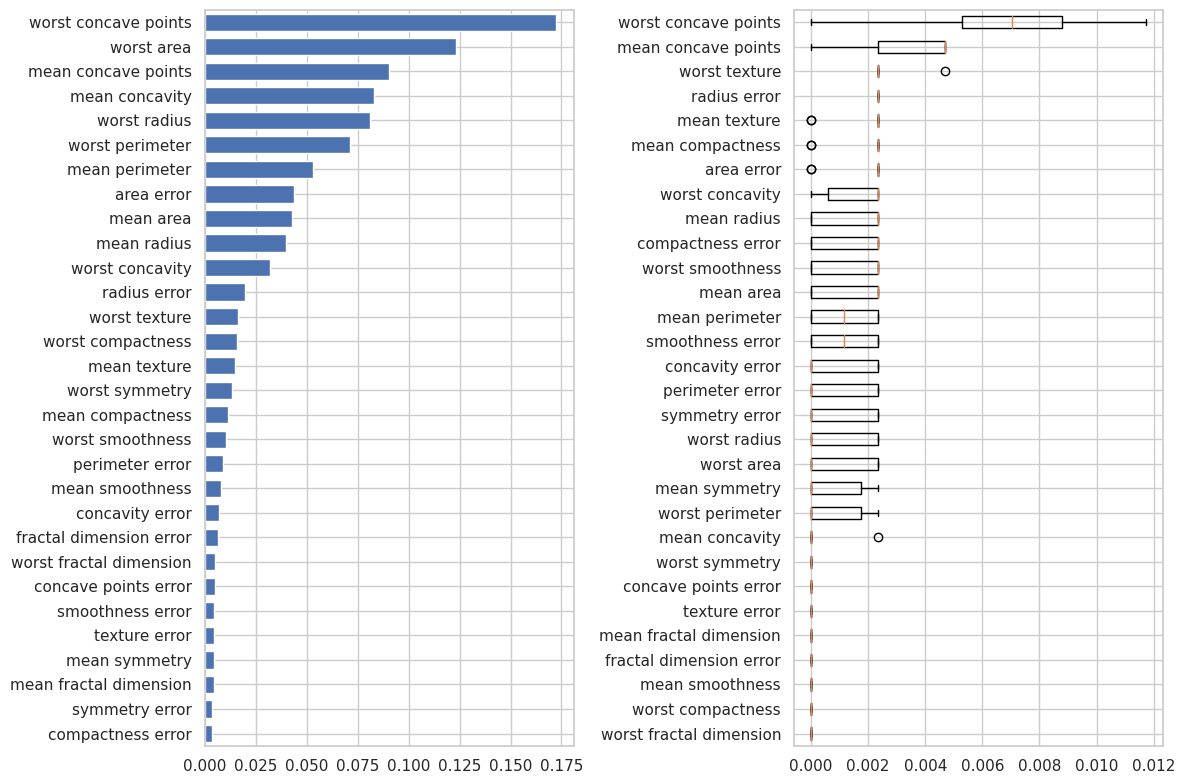
    vert=False,

    labels=data.feature\_names[perm\_sorted\_idx],

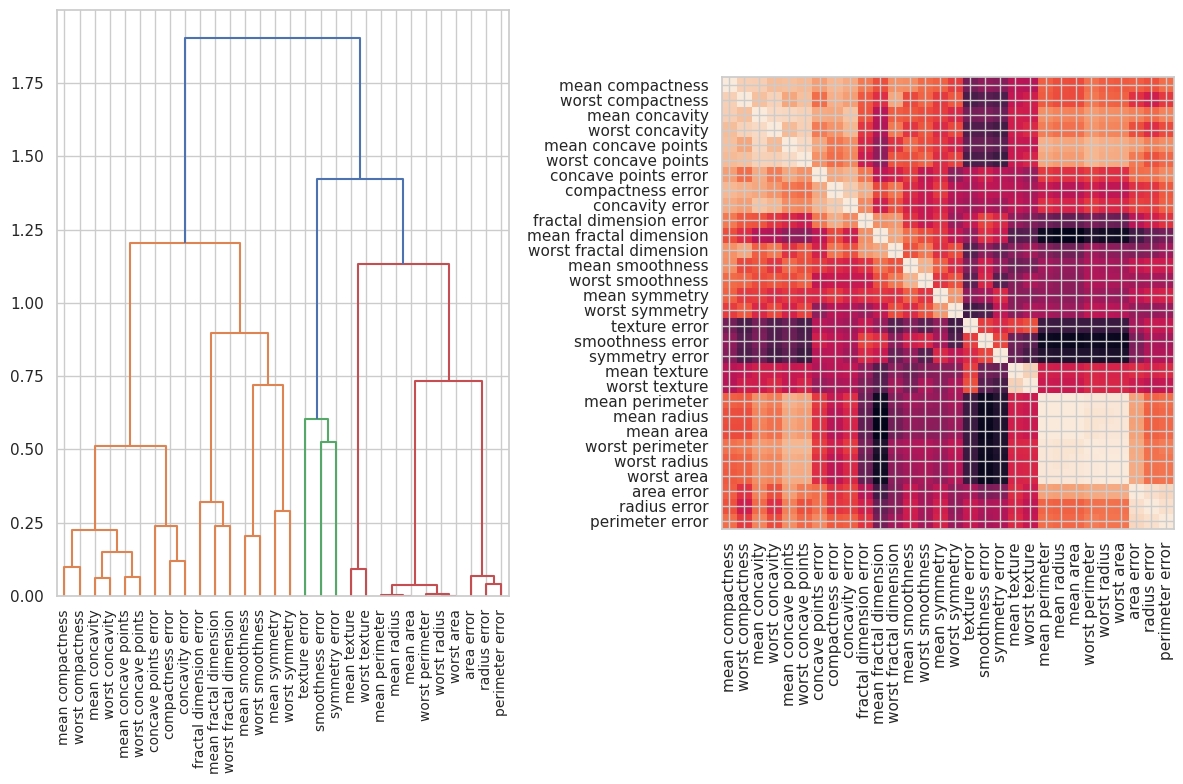
)

fig.tight\_layout()

plt.show()



Ketika fitur-fiturnya kolinear, permutasi satu fitur akan berdampak kecil pada kinerja model karena bisa mendapatkan informasi yang sama dari fitur yang dikorelasikan. Salah satu cara untuk menangani ini adalah dengan melakukan clustering.

Dari hasil clustering, fitur dipilih beberapa fitur yang paling berpengaruh dari setiap cluster. Dan dilakukan pengujian kembali sehingga diperoleh akurasinya adalah 0,97



Ketiga dengan menggunakan self training diperoleh

Chart, line chart

Description automatically generated

Contoh ini mengilustrasikan efek dari berbagai ambang pada self training. Dataset breast\_cancer dimuat, dan label dihapus sehingga hanya 50 dari 569 sampel yang memiliki label. SelfTrainingClassifier dipasang pada dataset ini, dengan ambang batas yang bervariasi. Gambar atas menunjukkan jumlah sampel berlabel yang dimiliki pengklasifikasi pada akhir fit, dan akurasi pengklasifikasi. Gambar yang bawah menunjukkan iterasi terakhir di mana sampel diberi label. Semua nilai divalidasi silang dengan 3 lipatan. Pada rentang [0.4, 0.5], pengklasifikasi belajar dari sampel yang diberi label dengan kepercayaan rendah. Sampel berkeyakinan rendah ini kemungkinan besar memiliki label prediksi yang salah, dan akibatnya, pemasangan label yang salah ini menghasilkan akurasi yang buruk. Perhatikan bahwa classifier melabeli hampir semua sampel, dan hanya membutuhkan satu iterasi.

Untuk ambang batas yang sangat tinggi dalam rentang [0.9, 1] diperoleh bahwa pengklasifikasi tidak menambah kumpulan datanya (jumlah sampel yang diberi label sendiri adalah 0). Akibatnya, akurasi yang dicapai dengan ambang 0,9999 sama dengan yang dicapai oleh pengklasifikasi terbimbing biasa.

Akurasi optimal terletak di antara kedua ekstrem ini pada ambang batas sekitar 0,7.

Dari ketiga model, model dengan akurasi terbaik adalah model random forest dengan akurasi 0,97.