Nama : Sayyid Abdullah

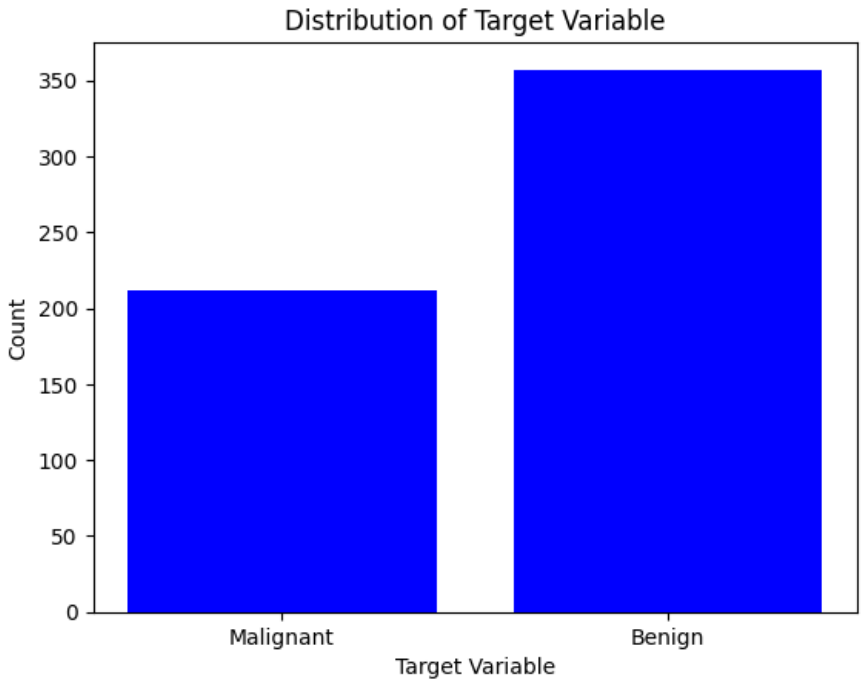
NPM : 2206130800

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.

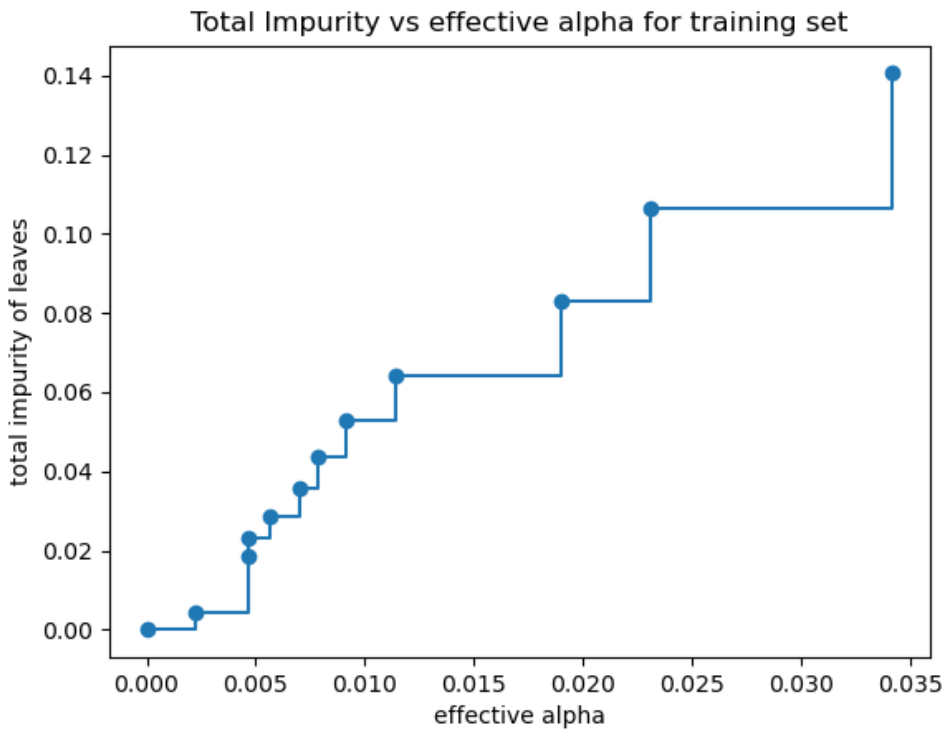
Petama menggunakan decision tree diperoleh

```
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}")
```

```
[1 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 0 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0
 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0
 1 1 1 0 1 1 0 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0
 1 1 0 1 0 1 1 1 0 1 1 1 0 1 0 0 1 1 0 0 0 1 1 1 0 1 1 1 0 1 0 1]
[1 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0
 1 0 1 1 0 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0
 1 1 1 0 1 1 0 1 1 0 0 0 0 0 1 1 1 1 0 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0
 1 1 0 1 0 1 1 1 0 0 1 1 0 1 0 0 1 1 0 0 0 0 1 1 0 0 1 1 0 1 0 1]
Accuracy: 0.951
```

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)
```

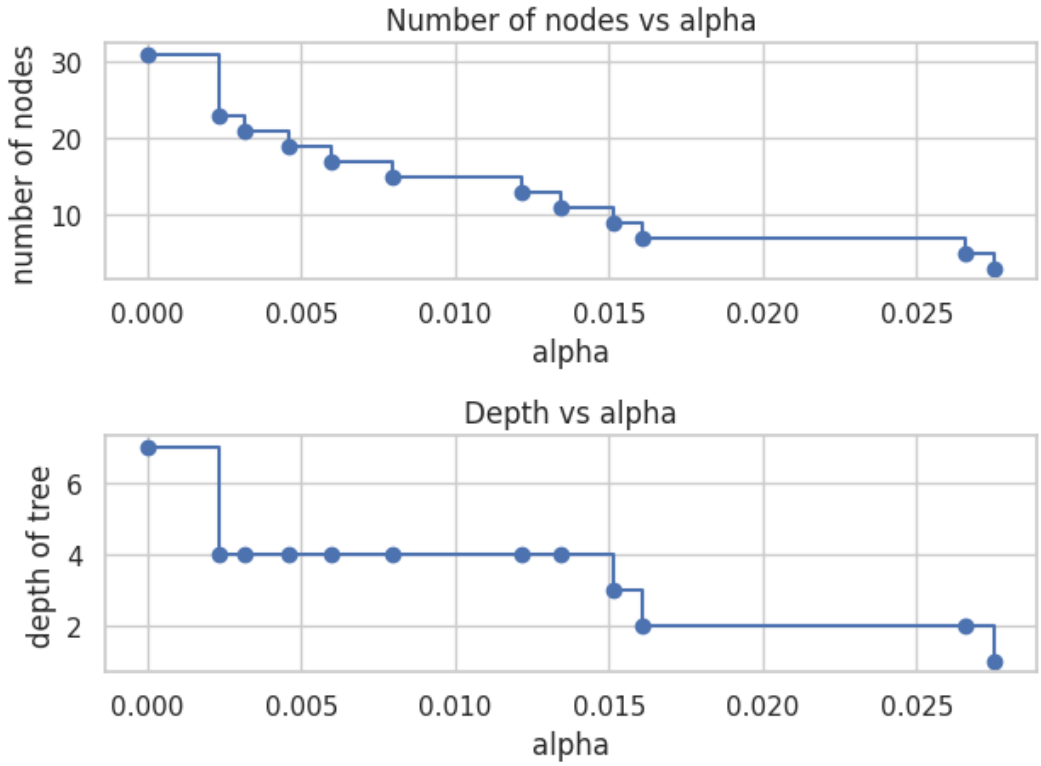
```
print(
    "Number of nodes in the last tree is: {} with ccp_alpha: {
}".format(
        clfs[-1].tree_.node_count, ccp_alphas[-1]
    )
)
```

Diperoleh output

```
Number of nodes in the last tree is: 1 with ccp_alpha: 0.3272984419327777
```

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="ste
ps-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, rand
om_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)))
```

```
Accuracy on test data: 0.97
```

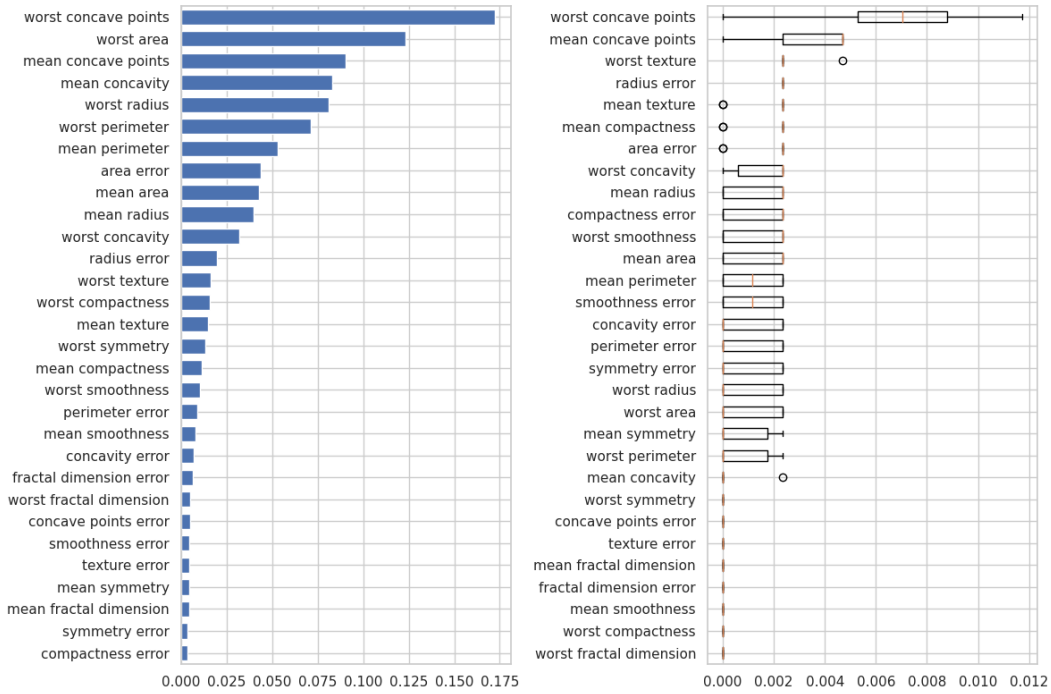
Diperoleh akurasi sebesar 0,97

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.

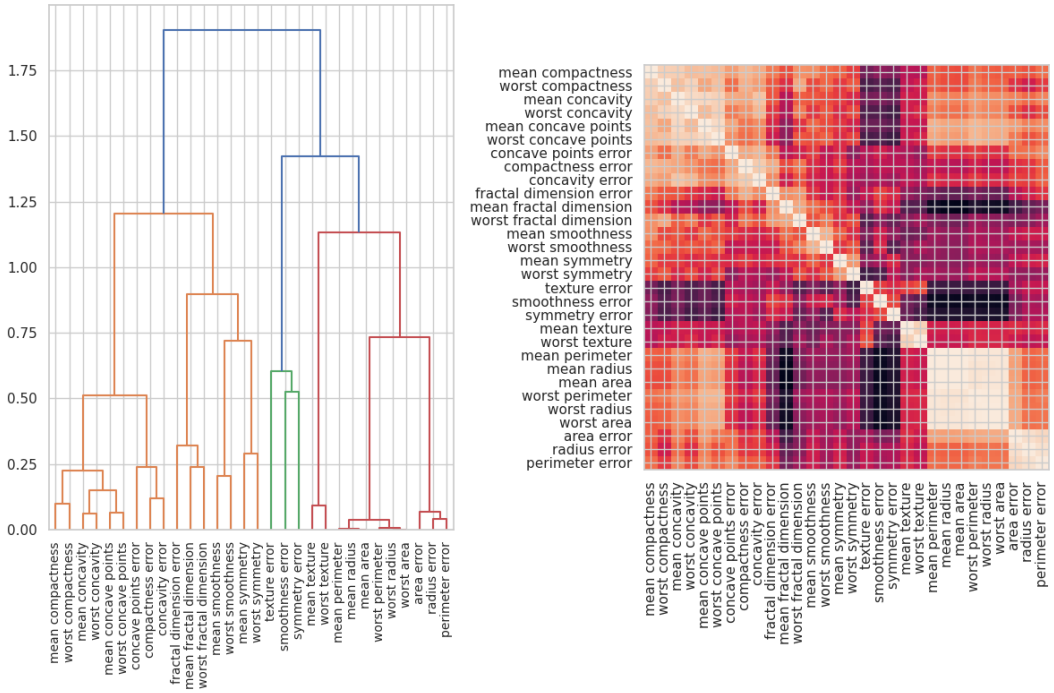
```
result = permutation_importance(clf, X_train, y_train, n_repea
ts=10, random_state=42)
perm_sorted_idx = result.importances_mean.argsort()

tree_importance_sorted_idx = np.argsort(clf.feature_importance
s_)
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_importanc
e_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.

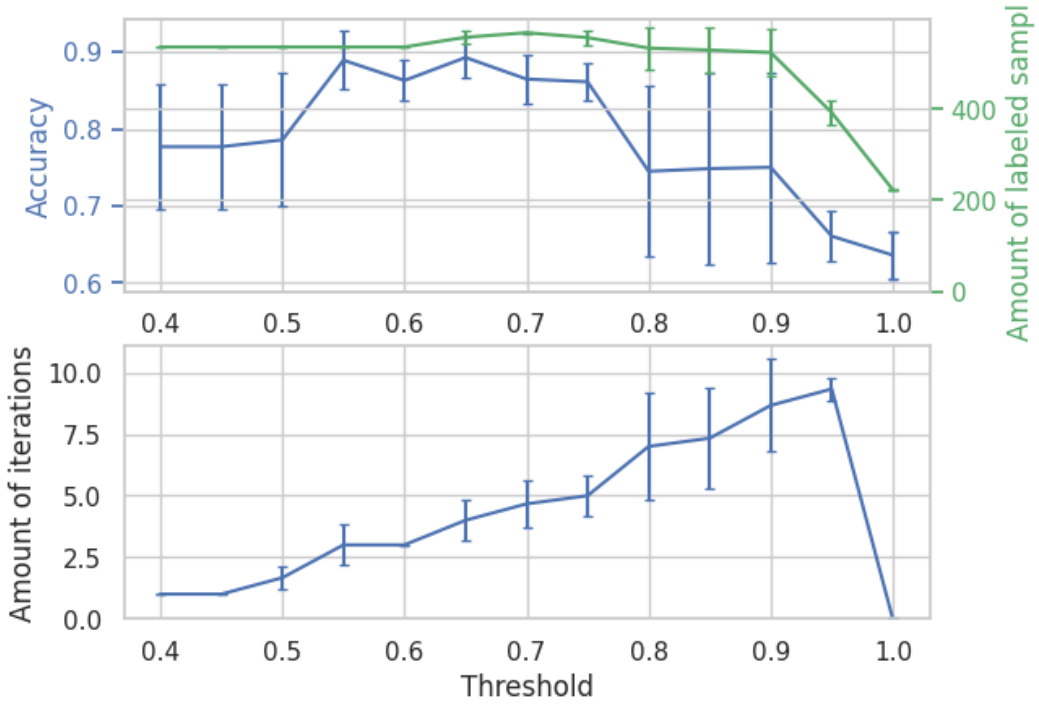


Dari hasil clustering, fitur dipilih beberapa fitur yang paling berpengaruh dari setiap

cluster. Dan dilakukan pengujian kembali sehingga diperoleh akurasinya adalah 0,97

```
Accuracy on test data with features removed: 0.97
```

Ketiga dengan menggunakan self training diperoleh



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.