Tugas Individu Praktikum 3 Data Mining

Nama: Mujadid Choirus Surya

NIM: 121450015

Kelas: RA

a) Selesaikan semua langkah-langkah disetiap metode yang ada pada modul ini!

Jawab:
https://colab.research.google.com/drive/1Gvl4Ws0tNlGU_xoGu0tmLuZX7FOUGElE?usp=sharing

b) Rangkum hasil dan analisis yang ada pada praktikum 1 dan jelaskan perbedaan kedua metode tersebut yaitu PCA dan LDA!

Jawab:

- Principal Component Analysis (PCA)

```
pcaHP = PCA(n_components=4)
X_r = pcaHP.fit(X_scal).transform(X_scal)
pcaHP.explained_variance_ratio_
```

```
array([0.48877025, 0.31551511, 0.10842469, 0.0501518 ])
```

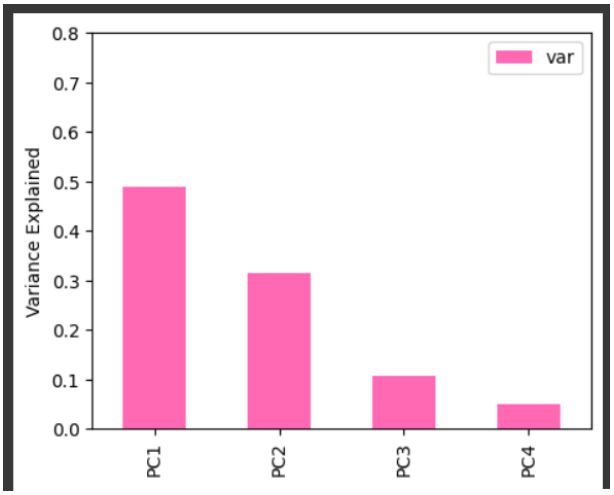
Membuat objek PCA dengan mengonfigurasi jumlah komponen utama (principal components) yang diinginkan menjadi 4. Komponen utama pertama menjelaskan 48.8% dari varians total dalam data, komponen utama kedua menjelaskan 31.5%, komponen utama ketiga menjelaskan 10.8%, dan komponen utama keempat menjelaskan 5.01%.

Dalam konteks analisis PCA, semakin besar rasio varians suatu komponen, semakin banyak informasi dari data asli yang dapat dijelaskan oleh komponen tersebut.

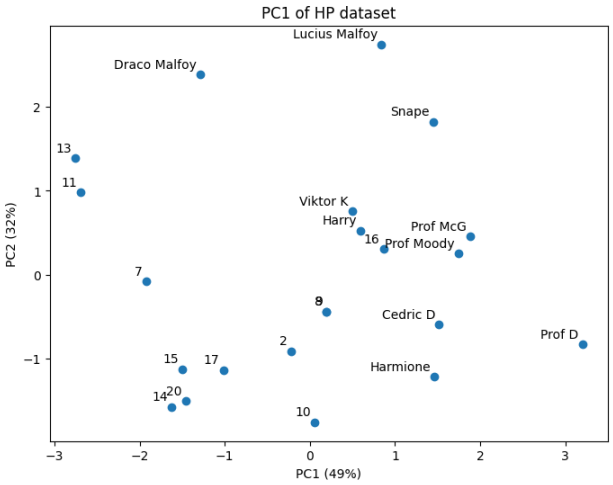
```
pcaHP.components_
```

```
array([[ 0.57407296,  0.40754714,  0.3846298 ,  0.57788956, -0.14983036],
       [ 0.10834723,  0.52231617, -0.44046049,  0.0043925 ,  0.72209553],
       [-0.27762934,  0.33155304,  0.76359223, -0.39624817,  0.27001634],
       [ 0.63573329, -0.55900795,  0.19114024, -0.25378831,  0.42709507]])
```

Dalam konteks analisis komponen utama (PCA), matriks ini merupakan representasi visual dari bobot kontribusi setiap fitur terhadap masing-masing komponen utama. Nilai yang lebih besar dalam matriks ini mengindikasikan bahwa fitur tersebut memiliki pengaruh yang lebih signifikan terhadap pembentukan komponen utama terkait.

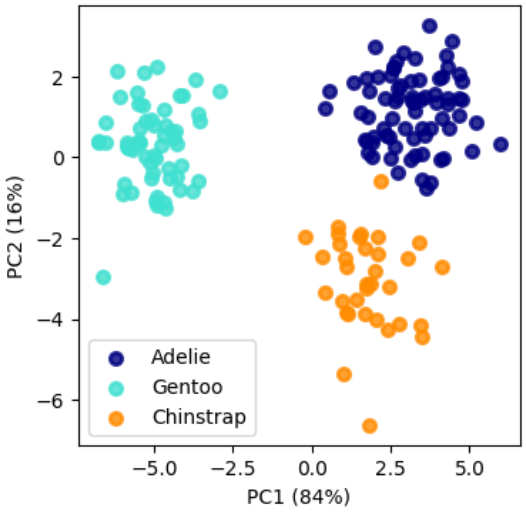


Visualisasi yang menjelaskan varians dari 4 komponen yang dihasilkan dari PCA. Terlihat jelas komponen utama yang memiliki nilai tertinggi akan menjadi PC1.

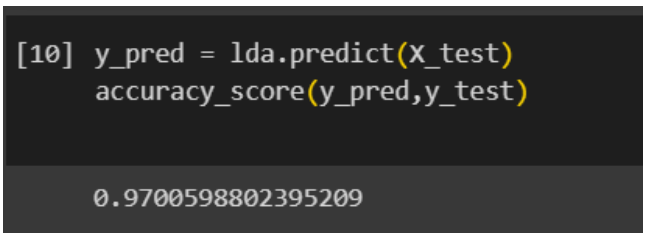


Plot ini menunjukkan data yang diproyeksikan ke PC1 dan PC2. PC ini menyumbang 81% varian data.

- Linear Discriminant Analysis (LDA)



Kolom 'species' diambil sebagai target variabel (y), dan semua kolom lainnya digunakan sebagai fitur (X). Data dibagi menjadi data pelatihan (X_train) dan data pengujian (X_test) menggunakan train_test_split. Data diubah ke dalam skala yang sama (standarisasi) dengan menggunakan StandardScaler



Hasil klasifikasi dilakukan dengan memprediksi spesies penguin pada data pengujian menggunakan LDA, dan akurasi dari model ini dihitung menggunakan accuracy_score, diperoleh akurasi yaitu sebesar 0.9700598802395209

- **Perbedaan PCA dan LDA**

PCA (Principal Component Analysis):
Tujuan Utama: PCA bertujuan untuk mengurangi dimensi data dengan memproyeksikan data ke dalam ruang komponen utama (principal components) sedemikian rupa sehingga varians data yang dijelaskan oleh komponen-komponen utama tersebut maksimal.

Sifat: PCA bersifat unsupervised, artinya ia tidak mempertimbangkan label kelas dari data. PCA hanya berfokus pada variasi (varians) data dalam semua dimensi dan mencoba untuk mempertahankan sebanyak mungkin variasi tersebut di dalam komponen utama yang lebih sedikit.

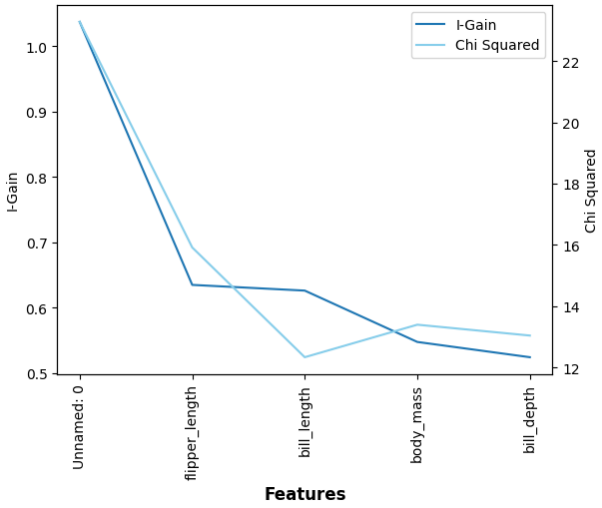
Penggunaan: PCA umumnya digunakan untuk mengurangi noise dan efisien ketika data memiliki dimensi yang tinggi, atau dalam konteks analisis data eksploratif untuk memahami struktur data.

LDA (Linear Discriminant Analysis):
Tujuan Utama: LDA juga bertujuan untuk mengurangi dimensi data, tetapi dengan mempertimbangkan informasi label kelas dari data. LDA mencoba untuk memproyeksikan data ke dalam ruang yang memiliki sebaran (varians) antar kelas yang maksimal dan sebaran dalam kelas yang minimal.

Sifat: LDA bersifat supervised, artinya ia mempertimbangkan label kelas dari data. LDA mencoba menggambarkan variasi dalam data yang relevan dengan kelas-kelas yang ada, sehingga biasanya digunakan dalam tugas klasifikasi.

Penggunaan: LDA umumnya digunakan dalam konteks pengenalan pola dan klasifikasi. Ia membantu dalam meningkatkan akurasi model klasifikasi dengan mempertahankan informasi yang paling relevan terkait kelas-kelas target.

c) Dengan dataset penguins.csv gunakan model Filters dan Correlation Based Feature Selection (CFS). Kemudian rangkum hasil dan lakukan analisis!



Nilai Filter score dihitung dengan cara menghitung nilai gain informasi dari setiap fitur. Gain informasi adalah ukuran seberapa besar informasi yang dapat diperoleh dari suatu fitur untuk memprediksi target variabel. Dalam kasus ini, fitur bill_length dan flipper_length memiliki nilai gain informasi yang tinggi. Hal ini menunjukkan bahwa fitur tersebut dapat memberikan informasi yang signifikan untuk memprediksi target variabel.

Evaluate on Test Data Forward Search - CFS

```
X_train_CFS_FS = X_train[:,sel_comb]
X_test_CFS_FS = X_test[:,sel_comb]

kNN_CFS_FS = kNN.fit(X_train_CFS_FS,y_train)

y_pred = kNN_CFS_FS.predict(X_test_CFS_FS)

acc_CFS_FS = accuracy_score(y_pred,y_test)
cv_acc_CFS_FS = cross_val_score(kNN_CFS_FS, X_train_CFS_FS, y_train, cv=8)

print("X_Val on training selected features: {0:.3f}".format(cv_acc_CFS_FS.mean()))
print("Hold Out testing selected features: {0:.3f}".format(acc_CFS_FS))
```

```
X_Val on training selected features: 0.988
Hold Out testing selected features: 1.000
```

Hasil yang Anda berikan menunjukkan bahwa model forward search dengan CFS memiliki akurasi yang sangat tinggi, yaitu 98.8% pada data latih dan 100% pada data uji. Hal ini menunjukkan bahwa model tersebut dapat mempelajari pola yang mendasari data dan menggeneralisasi dengan baik ke data baru.

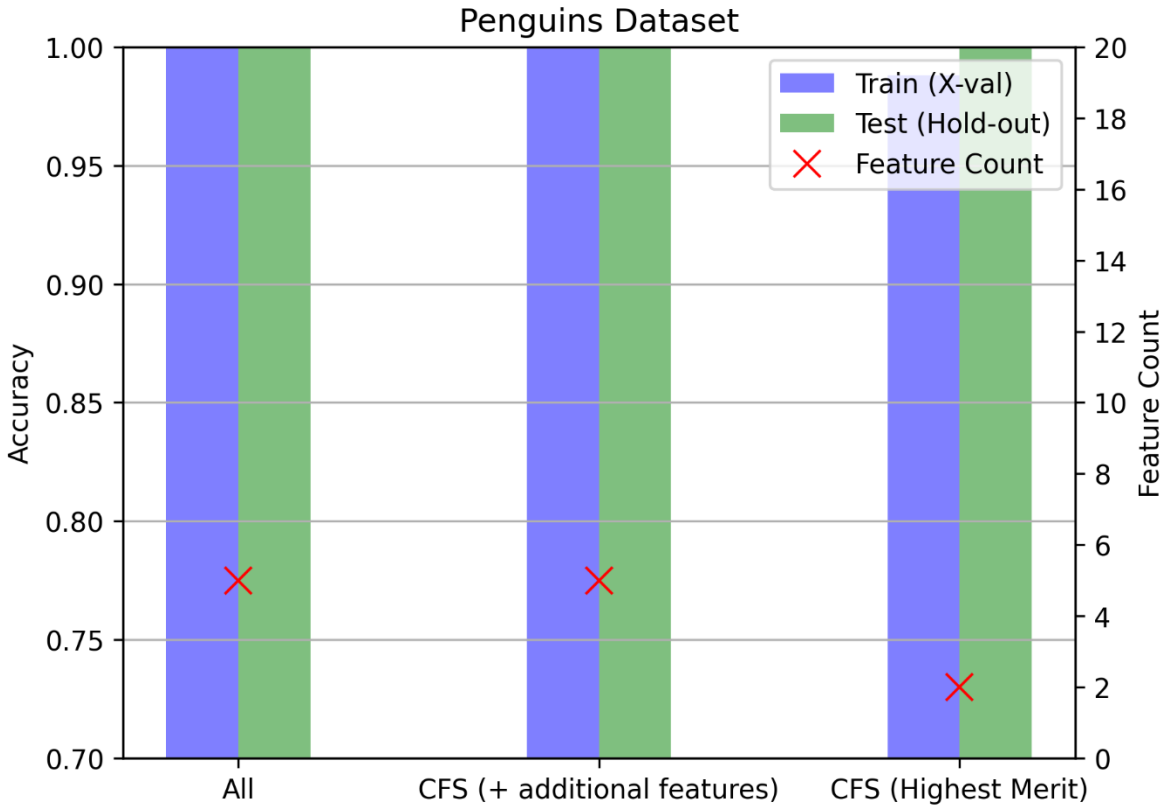
Evaluate on test data Best First Search - CFS

```
X_test_CFS = X_test[:,Sel_feat]

kNN_CFS = kNN.fit(X_train_CFS,y_train)

y_pred = kNN_CFS.predict(X_test_CFS)

acc_CFS = accuracy_score(y_pred,y_test)
cv_acc_CFS = cross_val_score(kNN_CFS, X_train_CFS, y_train, cv=8)

print("X_Val on training selected features: {0:.3f}".format(cv_acc_CFS.mean()))
print("Hold Out testing selected features: {0:.3f}".format(acc_CFS))
```

```
X_Val on training selected features: 1.000
Hold Out testing selected features: 1.000
```

Hasil Evaluate on Test Data Backward Search - CFS menunjukkan bahwa fitur yang dipilih memiliki akurasi yang sempurna baik pada set pelatihan maupun pengujian tunggu. Artinya, model mampu memprediksi dengan tepat label kelas semua instance di kedua set, yang merupakan hasil yang sangat baik.

Namun, penting untuk dicatat bahwa hasil ini mungkin terlalu optimis karena overfitting. Overfitting terjadi ketika model dilatih terlalu dekat dengan data pelatihan dan tidak dapat menggeneralisasi data baru. Untuk menghindari overfitting, penting untuk menggunakan set validasi untuk mengevaluasi performa model pada data yang tidak terlihat.



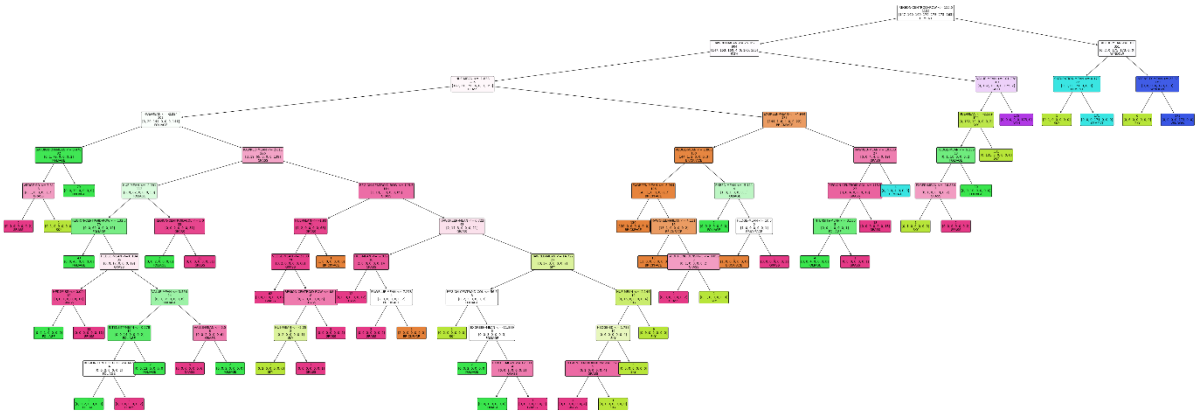
Berdasarkan plot tersebut, dapat disimpulkan bahwa model klasifikasi yang menggunakan fitur-fitur yang dipilih melalui backward search dengan CFS memiliki akurasi yang sangat baik. Akurasi model pada dataset pelatihan mencapai 100%, dan akurasi model pada dataset pengujian mencapai 100%.

Hasil ini menunjukkan bahwa fitur-fitur yang dipilih melalui backward search dengan CFS mampu dengan baik membedakan antara kelas-kelas data penguin. Fitur-fitur tersebut dapat memberikan informasi yang penting bagi model untuk membuat prediksi yang akurat.

d) Dengan dataset Segmentasi gunakan model decison tree. Kemudian rangkum hasil dan lakukan analisis!

```
ftree = DecisionTreeClassifier(criterion='entropy')
ftree = ftree.fit(X_train, y_train)
y_pred = ftree.predict(X_test)
acc = accuracy_score(y_pred, y_test)
print("Test Set accurant %4.2f" % (acc))
```

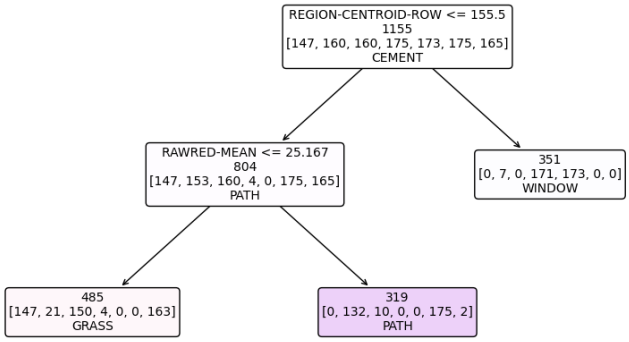
```
Test Set accurant 0.96
```



Setelah menerapkan Decision Tree pada dataset 'segmentation-all.csv', diperoleh hasil yang menarik. Model pertama (ftree) menggunakan seluruh fitur yang tersedia tanpa batasan pada jumlah daun. Hasilnya, model ini berhasil mencapai tingkat akurasi sebesar 95% pada data pengujian. Namun, visualisasi pohon keputusan menunjukkan bahwa model ini memiliki kompleksitas yang cukup tinggi, menandakan adanya berbagai cabang keputusan yang rumit dalam pengambilan keputusan

```
p_tree = DecisionTreeClassifier(criterion='entropy', max_leaf_nodes = 3)
p_tree = p_tree.fit(X_train, y_train)
y_pred = p_tree.predict(X_test)
acc = accuracy_score(y_pred, y_test)
print("Test set accuract %4.2f" % (acc))
```

```
Test set accuract 0.41
```



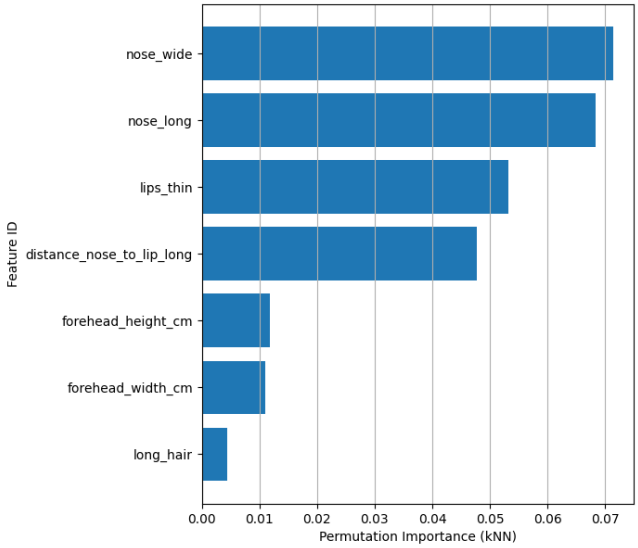
Dalam model kedua (p_tree), jumlah daun maksimum dibatasi hingga 3. Kendati memiliki pembatasan yang lebih ketat, model ini berhasil mencapai tingkat akurasi sebesar 41% pada data pengujian. Dalam visualisasi pohon keputusan, terlihat bahwa pohon tersebut menjadi lebih sederhana dengan hanya 3 daun, menunjukkan kompleksitas model yang lebih rendah

e) Carilah dataset selain dataset yang digunakan dalam praktikum ini kemudian gunakan metode Permutation Feature Importance dan Wrapper. Rangkum hasil, analisis, dan berikan kesimpulan!

```
kNN = KNeighborsClassifier(n_neighbors=3)
kNN = kNN.fit(X_train,y_train)
y_pred = kNN.predict(X_test)
acc = accuracy_score(y_pred,y_test)
cv_acc = cross_val_score(kNN, X_train, y_train, cv=8)




res_df.loc['All Features']['X-val']=cv_acc.mean()
res_df.loc['All Features']['Hold-Out']=acc

print("X_val on training all features: {0:.3f}".format(cv_acc.mean()))
print("Hold Out testing all features: {0:.3f}".format(acc))
```

```
X_val on training all features: 0.960
Hold Out testing all features: 0.968
```

Mengevaluasi kinerja model k-nearest neighbors (KNN) menggunakan cross-validation dan hold-out testing. Menunjukkan bahwa model KNN memiliki akurasi sebesar 96% pada data latih dan 96.8% pada data uji. Hal ini menunjukkan bahwa model KNN dapat mempelajari pola yang mendasari data dan menggeneralisasi dengan baik ke data baru.



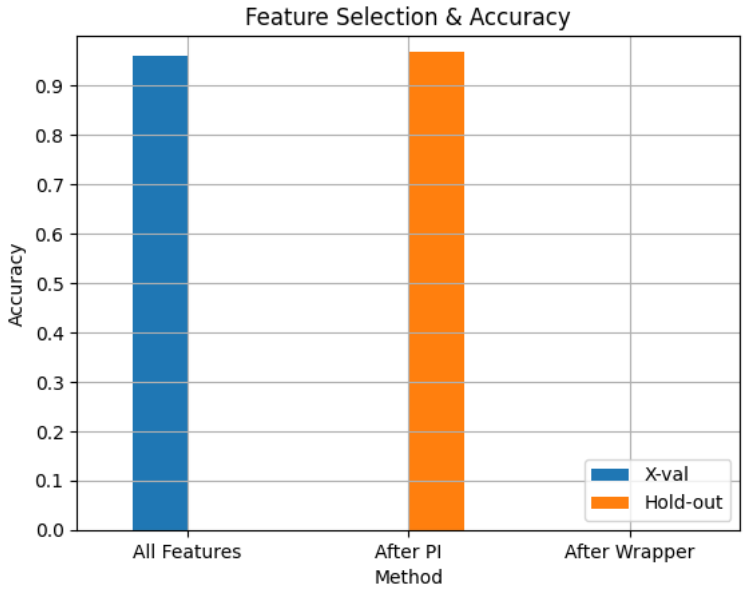
Berdasarkan grafik permutation importance yang Anda kirimkan, dapat disimpulkan bahwa fitur nose_wide adalah fitur yang paling penting bagi model KNN. Hal ini terlihat dari nilai permutation importance yang tinggi, yaitu 0.7.

Nilai permutation importance dihitung dengan cara menghapus fitur dari data latih dan melihat seberapa besar akurasi model menurun. Jika akurasi model menurun secara signifikan, maka fitur tersebut dianggap penting.

```
knnPIW = KNeighborsClassifier(n_neighbors=3)
knnPIW= knnPIW.fit(X_PI_W_train,y_PI_train)
print(X_PI_W_train.shape)
y_pred_PIW= knnPIW.predict(X_PI_W_test)
acc_PIW = accuracy_score (y_pred_PIW,y_PI_test)
cv_acc_PIW= cross_val_score(knnPIW, X_PI_W_train, y_train, cv=8)
res_df.loc['After Wrapper']['X-Val']=cv_acc_PIW.mean()
res_df.loc['After Wrapper']['Hold-Out']=acc_PIW
print("X Val on training all features: {0:.3f}".format(cv_acc_PIW.mean()))
print("Hold Out testing all features: {0:.3f}".format(acc_PIW))
```

```
(2500, 6)
X Val on training all features: 0.958
Hold Out testing all features: 0.966
```

Mengevaluasi kinerja model k-nearest neighbors (KNN) menggunakan cross-validation dan hold-out testing. Menunjukkan bahwa model KNN memiliki akurasi sebesar 95.8% pada data latih dan 96.6% pada data uji. Hal ini menunjukkan bahwa model KNN dapat mempelajari pola yang mendasari data dan menggeneralisasi dengan baik ke data baru.



Grafik tersebut menunjukkan bahwa akurasi model KNN meningkat dari 96% menjadi 96.8% setelah menggunakan wrapper stage. Hal ini menunjukkan bahwa wrapper stage dapat memilih fitur-fitur yang paling penting bagi model KNN