MODEL PREDIKSI KESELAMATAN PENUMPANG TITANIC (*TITANIC DATASET*) MENGGUNAKAN METODE REGRESI LOGISTIK, DECISION TREE, NAIVE BAYES DAN K-NEAREST NEIGHBOR

TUGAS UTS

OLEH:

ASEP RIDWAN HIDAYAT
231012050036

PROGRAM STUDI MAGISTER TEKNIK INFORMATIKA
PROGRAM PASCA SARJANA
UNIVERSITAS PAMULANG
TANGERANG SELATAN

2024

PREPROCESSING DATA

1. Deskrpsi Data

Variabel	Definisi	Keterangan
PassengerId	id penumpang	
Name	Nama penumpang	
Survival	Survival (Bertahan)	0 = Ya 1 = Tidak
Pclass	Ticket Class	1 = 1st, 2 = 2nd, 3 = 3rd
Sex		0 = male, 1 = female
Age	Usia dalam tahun	
Sibsp	saudara kandung/pasangan di kapal Titanic	Sibling = brother, sister, stepbrother, stepsister Spouse = husband, wife (mistresses and fiancés were ignored)
parch	orang tua/anak di kapal Titanic (family relations)	Parent = mother, father Child = daughter, son, stepdaughter, stepson parch=0
Ticket	Tiket	
fare	Tarif penumpang	
cabin	Nomor kabin	
embarked	Pelabuhan Embarkasi	C = Cherbourg, Q = Queenstown, S = Southampton

Tabel 1.1 : **https://www.kaggle.com/c/titanic/data**

Jumlah data training set yang diambil sebanyak 891 passenger dan jumlah test set sebanyak 1309 passenger, dengan jumlah variabel 12. Pengolahan data menggunakan Python

2. Data Preprocessing

Sebelum melakukan pengolahan dilakukan beberapa pengecekan jika data set null, dan beberapa kolom yang tidak perlu diikut sertakan, menentukan target data.

```
#Pengecekan data null
missing_values = train_data.isna().sum()
print(missing_values)
```

Kolom output data yang null

- Age - 177
- Cabin - 687
- Embarked – 2

Dari output diatas 3 kolom yang mempunyai data missing value (nilai null atau berisi Nan), dan akan dilakukan penanganan untuk ketiga variable tersebut.

Dari referensi yang didapat penangan nilai missing value bisa dilakukan dengan mengganti nilai tersebut dengan average (rata-rata) data, nilai median (nilai tengah) data atau mejadikan *dummys data* dengan 0 atau 1.

2.1 Menangai data Age

Untuk menangani nilai yang kosong pada age bisa dilakukan dengan menggantinya dengan nilai *median* dari age terhadap pclass, yaitu Pclass1 diisi dengan 37, Pclass2 diisi dengan 29 and PClass3 diisi dengan 24.

```
# menangani data age
def impute_train_age(cols):
    Age = cols.iloc[0]
    Pclass = cols.iloc[1]

    if pd.isnull(Age):
        if Pclass == 1:
            return 37
        elif Pclass == 2:
            return 29
        else:
            return 24
    else:
        return Age
train_data['Age'] = train_data[['Age','Pclass']].apply(im-
pute_train_age,axis=1)
```

2.2 Menangani data kolom cabin

Untuk menangani kolom cabin bisa dengan didrop, karena kolom cabin yang null sebanyak 687 dari 891.

```
train_data.drop(['Cabin',],axis=1,inplace=True)
```

2.3 Menangani data kolom Embarked

Sama halnya untuk kolom Embarked untuk nilai null dirubah menjadi nol atau satu, karena hanya 2 baris saja yang berisi null

```
train_data.dropna(inplace=True)
```

Setelah dilakukan beberapa pengecekan dan diberikan penanganan, dan selanjutnya merubah data categorical dengan data dummies dari library python.

```
train_data = pd.get_dummies(train_data, columns = ['Sex'],
drop_first=True)
train_data = pd.get_dummies(train_data,columns=['Em-
barked'],drop_first= True)
```

Ada beberapa kolom yang tidak diikut sertakan dalam pengolahan, seperti kolom id-pessanger, name , dan ticket. Berikut scriptnya

```
train_data.drop(['Name','Ticket','Passen-
gerId'],axis=1,inplace=True)
```

Setelah preprocessing data bisa dilakukan pengujian model Regresi logistic, Decision tree , naïve bayes dan K-NN

PROCESING DATA

REGRESI LOGISTIK

Berikut data script untuk pengolahan data regresi logistik untuk data set training passenger training dengan menggunakan python.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import RobustScaler
from sklearn.linear_model  import LogisticRegression

from sklearn.metrics import  confusion_matrix,classification_report, accu-
racy_score, f1_score, precision_score, recall_score, roc_auc_score,
roc_curve, log_loss, matthews_corrcoef
from sklearn import metrics

train_data= pd.read_csv('train_titanic_data_set.csv')

# PREPROCESING DATA
 missing_values = train_data.isna().sum()
# print(train_data.head())

# menangani data age
def impute_train_age(cols):
    Age = cols.iloc[0]
    Pclass = cols.iloc[1]

    if pd.isnull(Age):

        if Pclass == 1:
            return 37

        elif Pclass == 2:
            return 29

        else:
            return 24

    else:
        return Age
train_data ['Age'] = train_data 'Age','Pclass']].apply(im-
pute_train_age,axis=1)
```

```
# menangani data cabin
train_data.drop(['Cabin'],axis=1,inplace=True)
train_data.dropna(inplace=True)

# menangani data dummy untuk variable sex dan embarked
train_data = pd.get_dummies(train_data, columns = ['Sex'], drop_first=True)
train_data = pd.get_dummies(train_data,columns=['Embarked'],drop_first= True)

# Drop kolom yang tidak diikut sertakan
train_data.drop(['Name','Ticket','PassengerId'],axis=1,inplace=True)

# prosess pengolahan data
X = train_data.drop(['Survived'],axis = 1)
y = train_data['Survived']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,ran-
dom_state=101)

cols = X_train.columns

scaler = RobustScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

X_train = pd.DataFrame(X_train,columns=cols)
X_test = pd.DataFrame(X_test,columns=cols)

LogisticRegression_model = LogisticRegression(max_iter=4000)
LogisticRegression_model.fit(X_train,y_train)

y_pred = LogisticRegression_model.predict(X_test)
y_proba = LogisticRegression_model.predict_proba(X_test)

# Output
ca = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred, average='weighted')
precision = precision_score(y_test, y_pred, average='weighted', zero_divi-
sion=1) # Menangani kasus pembagian dengan nol
recall = recall_score(y_test, y_pred, average='weighted')
specificity = recall_score(y_test, y_pred, average='weighted')
logloss = log_loss(y_test, y_proba)
mcc = matthews_corrcoef(y_test, y_pred)

# mencari nilai kurva roc
fpr, tpr, thresholds = roc_curve(y_test, y_proba[:, 1])
```

```
plt.plot(fpr, tpr, label=name_classifiers)
roc_auc = roc_auc_score(y_test, y_proba[:, 1])

# output
name_classifiers = "Regresi Linear"
print(f"Model: {name_classifiers}")
print("Confusion Matrix:")
print(confusion_matrix)
print("Accuracy:", accuracy_score)
print("f1:",f1)
print("precision:", precision)
print("recall:", recall)
print("specificity:", specificity)
print("logloss:", logloss)
print("mcc:", mcc)
print("auroc:", roc_auc)

# print(classification_report(y_test, y_pred))

# confusion_matrix png
confusion_matrix = confusion_matrix(y_test,y_pred)

plt.figure(figsize=(10, 5))
sns.heatmap(confusion_matrix, annot=True, fmt='d', cmap='cool')
plt.show()

# grafik AOC
plt.figure()
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - ' + name_classifiers)
plt.legend(loc="lower right")
plt.show()
```

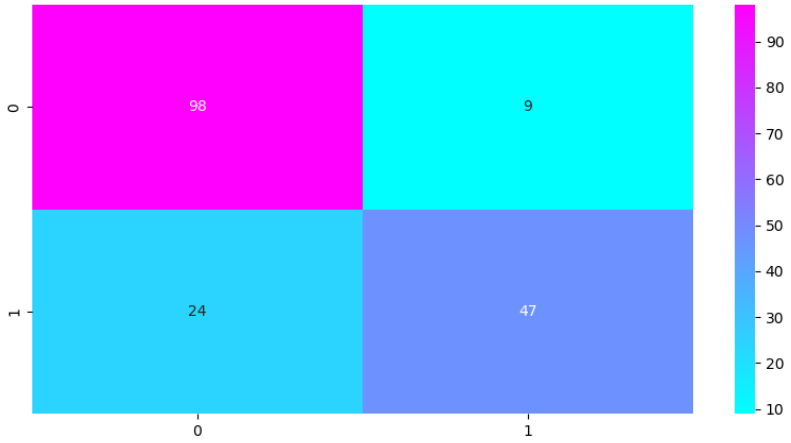
Dari pengolahan data untuk model Regresi Linear didapatkan nilai

1. **Confusion Matrik**

   Dibawah ini ouput data Matrik Confusion yang didapat dari model

col_0	0	1	Survived
0	98	9	
1	24	47	

Jika diplot kan dengan diagram visual, confusion matrik seperti dibawah ini:



Gambar 1.1 Tabel Confusion Matrik

Interpretasi dari Confusion matrik dibawah ini:

1) Kelas 0 (Predicitif Negatif dan Actual Negatif) artinya Terdapat 96 penumpang yang diprediksi dengan benar sebagai kelas 0 (penumpang tidak selamat (Not survive)), dan (Predictif Negatif dan Actual Positif) artinya 9 penumpang yang salah diprediksi sebagai kelas 0 (*Not survive Passanger/tidak selamat*) padahal actual nya selamat.
2) Kelas 1 (Predictif Positif dan Actual negatif): Ada 24 penumpang yang diprediksi dengan salah sebagai kelas 0 (penumpang selamat) padahal actualnya tidak selamat, dan 47 penumpang yang benar diprediksi kelas 1 (penumpang selamat) dan actual nya selamat.

Dari matriks confusion didapat nilai-nilai f1, Precision, Recall , Specificity.

2. **F1 Score: 0.809**

   Nilai F1 Score adalah rata-rata harmonik dari precision dan recall. F1 score mencapai nilai terbaik pada 1 dan terburuk pada 0.

   Dilihat dari hasil output data sebesar 0.80 ini bisa diartikan bahwa nilai rata-rata harmonik dari precision dan recall cukup baik

3. **Precision: 0.817**

Nilai precision atau presisi adalah proporsi dari hasil positif yang diidentifikasi dengan benar oleh model dari semua hasil positif yang diprediksi oleh model. Semakin tinggi nilainya, semakin sedikit hasil positif palsu yang dihasilkan oleh model.

Dari nilai yang didapat bisa diinterpretasikan hampir 81% hasil positif yang dihasilkan dari data

4. **Recall: 0.81**

   Recall adalah proporsi dari instance positif yang berhasil ditemukan dari semua instance positif yang aktual. Nilai yang tinggi menunjukkan bahwa model memiliki kemampuan yang baik dalam menemukan instance positif.

   Dari nilai yang didapat bisa diinterpretasikan hampir 81% (0.81) data menunjukkan bahwa model memiliki kemampuan yang baik.

5. **Specificity: 0.814**

   Nilai Specificity Ini adalah proporsi dari instance negatif yang diprediksi dengan benar dari semua instance negatif yang sebenarnya. Nilai yang tinggi menunjukkan bahwa model jarang membuat kesalahan dalam memprediksi instance positif sebagai negatif.

   Dari nilai yang didapat bisa diinterpretasikan hampir 81% (0.81) data model jarang membuat kesalahan dalam memprediksi instance positif sebagai negatif.

6. **Logloss: 0.415**

   Logloss Ini adalah pengukuran tingkat ketidakpastian dari prediksi model, di mana nilai yang rendah menunjukkan bahwa model memiliki tingkat kepercayaan yang tinggi dalam prediksi yang dibuatnya.

   Dari nilai yang didapat bisa diinterpretasikan data hanya memiliki 0.41 tingkat kepercayaan yang tinggi dalam prediksi yang dibuatnya.

7. **MCC (Matthews Corrcoef): 0.609**

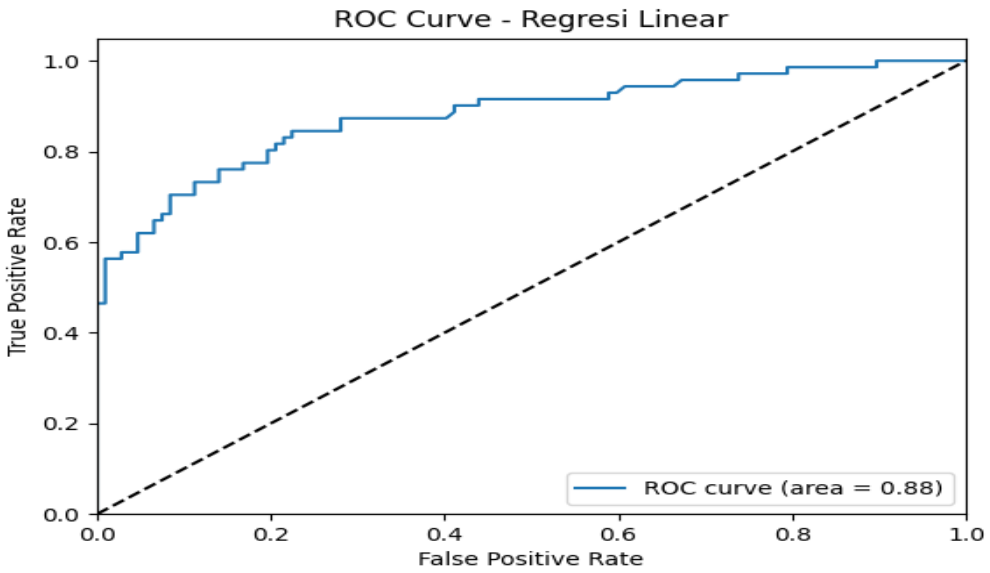
   MCC adalah pengukuran korelasi antara prediksi dan nilai observasi sebenarnya dari sebuah klasifikasi biner. Nilai mendekati 1 menunjukkan korelasi yang sempurna antara prediksi dan nilai sebenarnya.

   Dari nilai 0.61 (61%) artinya nilai mendekati 1 yang didapat bisa diinterpretasikan menunjukkan korelasi yang cukup bagus antara prediksi dan nilai sebenarnya.

8. **ROC : 0.88**

   ROC Curve adalah kurva yang memplot tingkat sensitivitas model terhadap tingkat 1-

spesifisitasnya. Area di bawah kurva ROC (Area Under the ROC Curve atau AUROC) juga sering digunakan sebagai metrik evaluasi, di mana nilai AUROC mendekati 1 menunjukkan kinerja yang lebih baik.



Gambar 1.2 Kurva ROC Regresi Linear

Dari nilai ROC **0.88** artinya nilai mendekati 1 yang didapat bisa diinterpretasikan mendekati 1 menunjukkan kinerja yang lebih baik. Dan dapat dilihat dari grafik pun cenderung mendekati 1.

PROCESING DATA

DECISION TREE

Berikut data scrip untuk pengolahan data Decision Tree untuk data set training passenger training dengan menggunakan python.

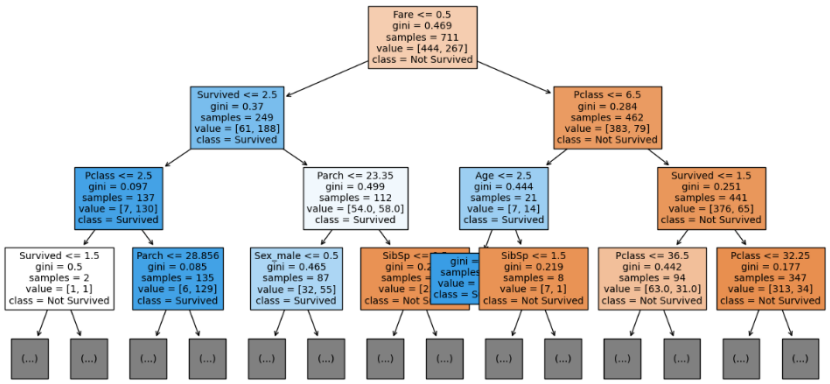
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import RobustScaler
from sklearn.linear_model  import LogisticRegression
from sklearn.metrics import  confusion_matrix,classification_re-
port, accuracy_score, f1_score, precision_score, recall_score,
roc_auc_score, roc_curve, log_loss, matthews_corrcoef
from sklearn import metrics, tree
from sklearn.tree import plot_tree
train_data = pd.read_csv('train.csv')

# print(df.head())
# preprosesin data
# menangani data age
def impute_train_age(cols):
    Age = cols.iloc[0]
    Pclass = cols.iloc[1]

    if pd.isnull(Age):

        if Pclass == 1:
            return 37

        elif Pclass == 2:
            return 29

        else:
            return 24

    else:
        return Age

train_data['Age'] = train_data[['Age','Pclass']].apply(im-
pute_train_age,axis=1)

train_data.drop(['Cabin'],axis=1,inplace=True)
```

```
train_data.dropna(inplace=True)

# drop kolom yang tidak diperlukan
train_data.drop(['Name','Ticket','Passen-
gerId'],axis=1,inplace=True)

# MERUBAH VARIABEL CATEGORICAL DAUMMY INT
train_data = pd.get_dummies(train_data, columns = ['Sex'],
drop_first=True)
train_data = pd.get_dummies(train_data,columns=['Em-
barked'],drop_first= True)

# tentukan targetd data
X = train_data.drop(['Survived'],axis = 1)
y = train_data['Survived']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2,random_state=0)

classifier = tree.DecisionTreeClassifier()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
y_proba = classifier.predict_proba(X_test)

confusion_matrix = confusion_matrix(y_test,y_pred)

ca = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred, average='weighted')
precision = precision_score(y_test, y_pred, average='weighted',
zero_division=1) # Menangani kasus pembagian dengan nol
recall = recall_score(y_test, y_pred, average='weighted')
specificity = recall_score(y_test, y_pred, average='weighted')
logloss = log_loss(y_test, y_proba)
mcc = matthews_corrcoef(y_test, y_pred)
name_classifiers = "Decision tree"
# mencari nilai kurva roc
fpr, tpr, thresholds = roc_curve(y_test, y_proba[:, 1])
plt.plot(fpr, tpr, label=name_classifiers)
roc_auc = roc_auc_score(y_test, y_proba[:, 1])

# output

print(f"Model: {name_classifiers}")
print("Confusion Matrix:")
print(confusion_matrix)
print("Accuracy:", accuracy_score)
print("f1:",f1)
```

```
print("precision:", precision)
print("recall:", recall)
print("specificity:", specificity)
print("logloss:", logloss)
print("mcc:", mcc)
print("auroc:", roc_auc)

# # confusion_matrix png
plt.figure(figsize=(10, 5))
sns.heatmap(confusion_matrix, annot=True, fmt='d', cmap='cool')
plt.show()

# # grafik AOC
plt.figure()
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - ' + name_classifiers)
plt.legend(loc="lower right")
plt.show()

# matrik confusion jadi png
plt.figure(figsize=(15,10))
plot_tree(classifier, feature_names=train_data.columns,
class_names=['Not Survived', 'Survived'], filled=True,max_depth=5,
fontsize=5)
plt.show()
```

1. **Diagram Pohon Klasifikasi**

   Berikut didapatkan diagram pohon yang hanya samapi 5 node,karena jika ditampilkan data begitu banyak:



Gambar 1.3 Grafik Pohon Klasifikasi

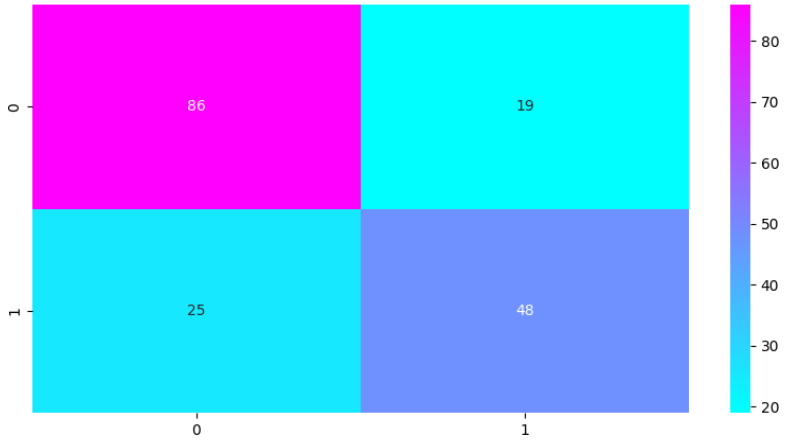
Jika dilihat dari pohon klasifikasi variable yang pertama diklasifikasikan adalah berdasarkan fare (tarif penumpang), jika nilai fare <= 0.5 maka data dikklasifikasikan 1 artinya Not Survived, dan jika fare nya lebih besar dari 0.5 maka data selanjutnya diklasifikan sebagai 0 atau survived, selanjutnya setelah variable fare yang <0.5 adalah variabel Pclass <=6.5 data diklasifikasikan 1 jika sebalikanya diklasifikan 0 dan seterusnya sesuai diagram pohon diatas.

2. Confusion Matriks

Output data seperti ini :

col_0	0	1
Survived		
0	87	18
1	24	49

Jika dtampilak dengan visual plot seperti dibawah ini:



Gambar 1.4 Tabel Confusion Matrik

Interpretasi dari Confusion matrik untuk decision tree dibawah ini:

1) Kelas 0 (Predicitif Negatif dan Actual Negatif) artinya Terdapat 86 penumpang yang diprediksi dengan benar sebagai kelas 0 (penumpang tidak selamat (Not survive)), dan (Predictif Negatif dan Actual Positif) artinya 19 penumpang yang salah diprediksi sebagai kelas 0 (*Not survive Passanger/tidak selamat*) padahal actual nya selamat.
2) Kelas 1 (Predictif Positif dan Actual negatif): Ada 25 penumpang yang diprediksi dengan salah sebagai kelas 0 (penumpang selamat) padahal actualnya tidak selamat, dan 48 penumpang yang benar diprediksi kelas 1 (penumpang selamat) dan actual nya selamat.

3. **F1 score: 0.75**
   Nilai F1 Score adalah rata-rata harmonik dari precision dan recall. F1 score mencapai nilai terbaik pada 1 dan terburuk pada 0.
   Dilihat dari hasil output data sebesar 0.75 ini bisa diartikan bahwa nilai rata-rata harmonik dari precision dan recall cukup baik
4. **Precision: 0.7508**
   Nilai precision atau presisi adalah proporsi dari hasil positif yang diidentifikasi dengan benar oleh model dari semua hasil positif yang diprediksi oleh model. Semakin tinggi nilainya, semakin sedikit hasil positif palsu yang dihasilkan oleh model.
   Dari nilai yang didapat bisa diinterpretasikan hampir 75% hasil positif yang dihasilkan dari data

5. **Recall: 0.75280**

   Recall adalah proporsi dari instance positif yang berhasil ditemukan dari semua instance positif yang aktual. Nilai yang tinggi menunjukkan bahwa model memiliki kemampuan yang baik dalam menemukan instance positif.

   Dari nilai yang didapat bisa diinterpretasikan hampir 75% (0.75) data menunjukkan bahwa model memiliki kemampuan yang baik.

6. **Specificity: 0.752808**

   Nilai Specificity Ini adalah proporsi dari instance negatif yang diprediksi dengan benar dari semua instance negatif yang sebenarnya. Nilai yang tinggi menunjukkan bahwa model jarang membuat kesalahan dalam memprediksi instance positif sebagai negatif.

   Dari nilai yang didapat bisa diinterpretasikan hampir 75% (0.75) data model jarang membuat kesalahan dalam memprediksi instance positif sebagai negatif.

7. **Logloss : 8.533**

   Logloss Ini adalah pengukuran tingkat ketidakpastian dari prediksi model, di mana nilai yang rendah menunjukkan bahwa model memiliki tingkat kepercayaan yang tinggi dalam prediksi yang dibuatnya.

   Dari nilai yang didapat bisa diinterpretasikan data hanya memiliki 8.5 tingkat kepercayaan yang kurang tinggi dalam prediksi yang dibuatnya.

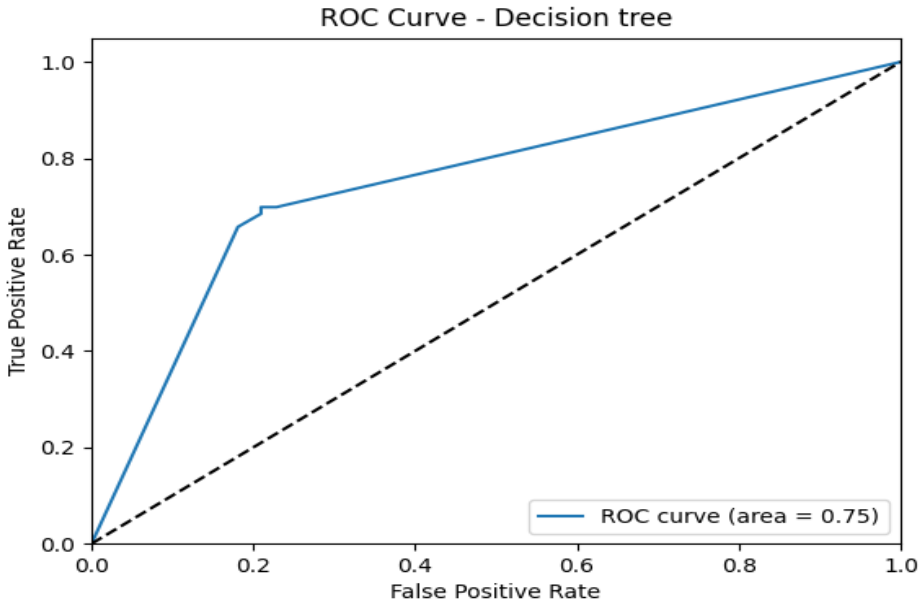
8. **Mcc (Matthews Corrcoef): 0.483**

   MCC adalah pengukuran korelasi antara prediksi dan nilai observasi sebenarnya dari sebuah klasifikasi biner. Nilai mendekati 1 menunjukkan korelasi yang sempurna antara prediksi dan nilai sebenarnya.

   Dari nilai 0.48 artinya nilai jauh dari nilai 1, bisa diinterpretasikan menunjukkan korelasi yang kurang bagus antara prediksi dan nilai sebenarnya.

9. **Auroc: 0.7471624266144814**

   ROC Curve adalah kurva yang memplot tingkat sensitivitas model terhadap tingkat 1-spesifisitasnya. Area di bawah kurva ROC (Area Under the ROC Curve atau AUROC) juga sering digunakan sebagai metrik evaluasi, di mana nilai AUROC mendekati 1 menunjukkan kinerja yang lebih baik.



Gambar 1.6 ROC Curve Decision Tree

Dari nilai ROC **0.74** artinya nilai mendekati 1 yang didapat bisa diinterpretasikan mendekati 1 menunjukkan kinerja yang lebih baik. Dan dapat dilihat dari grafik pun cenderung mendekati 1.

PROCESING DATA

NAIVE BAYES

Berikut data scrip untuk pengolahan data Naive Bayes untuk data set training passenger training dengan menggunakan python.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import  confusion_matrix, accuracy_score,
f1_score, precision_score, recall_score, roc_auc_score, roc_curve,
log_loss, matthews_corrcoef
from sklearn.naive_bayes import GaussianNB

train_data = pd.read_csv('train.csv')
# test_data = pd.read_csv('test.csv')
# print(df.head())

# menangani data age
def impute_train_age(cols):
    Age = cols.iloc[0]
    Pclass = cols.iloc[1]

    if pd.isnull(Age):

        if Pclass == 1:
            return 37

        elif Pclass == 2:
            return 29

        else:
            return 24

    else:
        return Age

train_data['Age'] = train_data[['Age','Pclass']].apply(im-
pute_train_age,axis=1)

train_data.drop(['Cabin'],axis=1,inplace=True)
train_data.dropna(inplace=True)
```

```
train_data = pd.get_dummies(train_data, columns = ['Sex'],
drop_first=True)
train_data = pd.get_dummies(train_data,columns=['Em-
barked'],drop_first= True)

train_data.drop(['Name','Ticket','Passen-
gerId'],axis=1,inplace=True)

X = train_data.drop(['Survived'],axis = 1)
y = train_data['Survived']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2,random_state=0)

classifier = GaussianNB()

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
y_proba = classifier.predict_proba(X_test)

confusion_matrix = confusion_matrix(y_test,y_pred)
# print(confusion_matrix)
confusion_matrix_prob =pd.crosstab(y_test,y_pred)

ca = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred, average='weighted')
precision = precision_score(y_test, y_pred, average='weighted',
zero_division=1) # Menangani kasus pembagian dengan nol
recall = recall_score(y_test, y_pred, average='weighted')
specificity = recall_score(y_test, y_pred, average='weighted')
logloss = log_loss(y_test, y_proba)
mcc = matthews_corrcoef(y_test, y_pred)
name_classifiers = "Naive Bayes"
# mencari nilai kurva roc
fpr, tpr, thresholds = roc_curve(y_test, y_proba[:, 1])
plt.plot(fpr, tpr, label=name_classifiers)
roc_auc = roc_auc_score(y_test, y_proba[:, 1])

# output
print(f"Model: {name_classifiers}")
print("Confusion Matrix:")
print(confusion_matrix)
print("Accuracy:", accuracy_score)
print("f1:",f1)
print("precision:", precision)
print("recall:", recall)
```

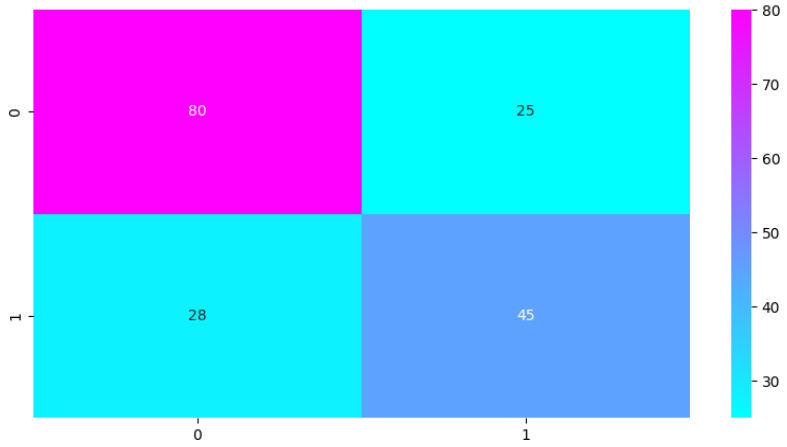
```
print("specificity:", specificity)
print("logloss:", logloss)
print("mcc:", mcc)
print("auroc:", roc_auc)

# confusion_matrix png
plt.figure(figsize=(10, 5))
sns.heatmap(confusion_matrix, annot=True, fmt='d', cmap='cool')
plt.show()

## grafik AOC
plt.figure()
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - ' + name_classifiers)
plt.legend(loc="lower right")
plt.show()
```

1. **Confusion Matriks**

Output data seperti ini

```
col_0     0   1
Survived
0        80  25
1        28  45
Model: Naive Bayes
```

Jika dtampilak dengan visual plot seperti dibawah ini:



Gambar 1.7 Tabel Confusion Matrik

Interpretasi dari Confusion matrik untuk decision tree dibawah ini:

1) Kelas 0 (Predicitif Negatif dan Actual Negatif) artinya Terdapat 80 penumpang yang diprediksi dengan benar sebagai kelas 0 (penumpang tidak selamat (Not survive)), dan (Predictif Negatif dan Actual Positif) artinya 25 penumpang yang salah diprediksi sebagai kelas 0 (*Not survive Passanger/tidak selamat*) padahal actual nya selamat.
2) Kelas 1 (Predictif Positif dan Actual negatif): Ada 28 penumpang yang diprediksi dengan salah sebagai kelas 0 (penumpang selamat) padahal actualnya tidak selamat, dan 45 penumpang yang benar diprediksi kelas 1 (penumpang selamat) dan actual nya selamat.

2. **F1 score: 0.70**
   Nilai F1 Score adalah rata-rata harmonik dari precision dan recall. F1 score mencapai nilai terbaik pada 1 dan terburuk pada 0.

   Dilihat dari hasil output data sebesar 0.70 ini bisa diartikan bahwa nilai rata-rata harmonik dari precision dan recall cukup baik

3. **Precision: 0.70**
   Nilai precision atau presisi adalah proporsi dari hasil positif yang diidentifikasi dengan benar oleh model dari semua hasil positif yang diprediksi oleh model. Semakin tinggi nilainya, semakin sedikit hasil positif palsu yang dihasilkan oleh model.

   Dari nilai yang didapat bisa diinterpretasikan hampir 70% hasil positif yang dihasilkan dari data

4. **Recall: 0.702247191011236**

   Recall adalah proporsi dari instance positif yang berhasil ditemukan dari semua instance positif yang aktual. Nilai yang tinggi menunjukkan bahwa model memiliki kemampuan yang baik dalam menemukan instance positif.

   Dari nilai yang didapat bisa diinterpretasikan hampir 70% (0.70) data menunjukkan bahwa model memiliki kemampuan yang baik.

5. **Specificity: 0.70**

   Nilai Specificity Ini adalah proporsi dari instance negatif yang diprediksi dengan benar dari semua instance negatif yang sebenarnya. Nilai yang tinggi menunjukkan bahwa model jarang membuat kesalahan dalam memprediksi instance positif sebagai negatif.

   Dari nilai yang didapat bisa diinterpretasikan hampir 70% (0.70) data model jarang membuat kesalahan dalam memprediksi instance positif sebagai negatif.

6. **Logloss: 0.91**

   Logloss Ini adalah pengukuran tingkat ketidakpastian dari prediksi model, di mana nilai yang rendah menunjukkan bahwa model memiliki tingkat kepercayaan yang tinggi dalam prediksi yang dibuatnya.

   Dari nilai yang didapat bisa diinterpretasikan data hanya memiliki 0.91 tingkat kepercayaan tinggi dalam prediksi yang dibuatnya.

7. **MCC (Matthews Corrcoef): 0.380961440785217**

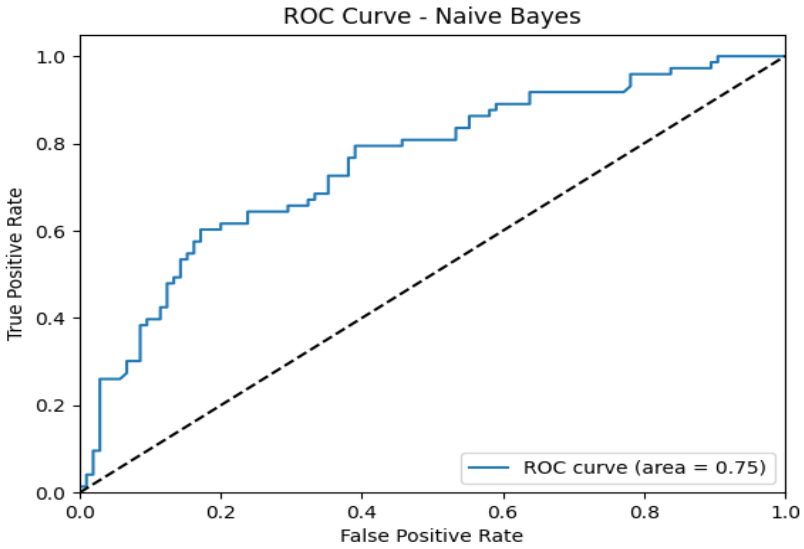
   MCC adalah pengukuran korelasi antara prediksi dan nilai observasi sebenarnya dari sebuah klasifikasi biner. Nilai mendekati 1 menunjukkan korelasi yang sempurna antara prediksi dan nilai sebenarnya.

   Dari nilai 0.38 artinya nilai jauh dari nilai 1, bisa diinterpretasikan menunjukkan korelasi yang kurang bagus antara prediksi dan nilai sebenarnya.

8. **ROC Curve: 0.7532941943900848**

   ROC Curve adalah kurva yang memplot tingkat sensitivitas model terhadap tingkat 1-spesifisitasnya. Area di bawah kurva ROC (Area Under the ROC Curve atau AUROC) juga sering digunakan sebagai metrik evaluasi, di mana nilai AUROC mendekati 1 menunjukkan kinerja yang lebih baik.

   Berikut grafik yang ditampilkan dari output data yang didapat.



Gambar 1.8 Grafik ROC Naive Bayes

Dari nilai ROC **0.75** artinya nilai mendekati 1 yang didapat bisa diinterpretasikan mendekati 1 menunjukkan kinerja yang lebih baik. Dan dapat dilihat dari grafik pun cenderung mendekati 1.

PROCESING DATA

K-NEAREST NEIGHBOR

Berikut data scrip untuk pengolahan data K-Nearest Neighbour untuk data set training passenger training dengan menggunakan python.

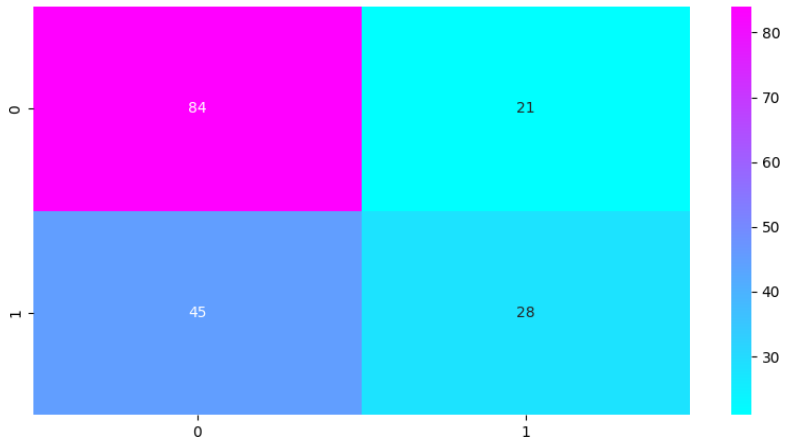
```
from sklearn.preprocessing import LabelEncoder
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score,
f1_score, precision_score, recall_score, roc_auc_score, roc_curve,
log_loss, matthews_corrcoef

train_data = pd.read_csv('train.csv')

# Preporesing data
# menangani data age
def impute_train_age(cols):
    Age = cols.iloc[0]
    Pclass = cols.iloc[1]

    if pd.isnull(Age):

        if Pclass == 1:
            return 37

        elif Pclass == 2:
            return 29

        else:
            return 24

    else:
        return Age

train_data['Age'] = train_data[['Age','Pclass']].apply(im-
pute_train_age,axis=1)

train_data.drop(['Cabin'],axis=1,inplace=True)
train_data.dropna(inplace=True)
```

```
train_data = pd.get_dummies(train_data, columns = ['Sex'],
drop_first=True)
train_data = pd.get_dummies(train_data,columns=['Em-
barked'],drop_first= True)

train_data.drop(['Name','Ticket','Passen-
gerId'],axis=1,inplace=True)

X = train_data.drop(['Survived'],axis = 1)
y = train_data['Survived']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2,random_state=0)

classifier = KNeighborsClassifier(n_neighbors=31)

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
y_proba = classifier.predict_proba(X_test)

confusion_matrix = confusion_matrix(y_test,y_pred)
# print(confusion_matrix)
confusion_matrix_prob =pd.crosstab(y_test,y_pred)
print(confusion_matrix_prob)
print('Y predi')
print(y_pred)

ca = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred, average='weighted')
precision = precision_score(y_test, y_pred, average='weighted',
zero_division=1) # Menangani kasus pembagian dengan nol
recall = recall_score(y_test, y_pred, average='weighted')
specificity = recall_score(y_test, y_pred, average='weighted')
logloss = log_loss(y_test, y_proba)
mcc = matthews_corrcoef(y_test, y_pred)
name_classifiers = "K-Nearest Neighbor K = 31"
# mencari nilai kurva roc
fpr, tpr, thresholds = roc_curve(y_test, y_proba[:, 1])
plt.plot(fpr, tpr, label=name_classifiers)
roc_auc = roc_auc_score(y_test, y_proba[:, 1])

# output

print(f"Model: {name_classifiers}")
print("Confusion Matrix:")
print(confusion_matrix)
```

```
print("Accuracy:", accuracy_score)
print("f1:",f1)
print("precision:", precision)
print("recall:", recall)
print("specificity:", specificity)
print("logloss:", logloss)
print("mcc:", mcc)
print("auroc:", roc_auc)

# confusion_matrix png
plt.figure(figsize=(10, 5))
sns.heatmap(confusion_matrix, annot=True, fmt='d', cmap='cool')
plt.show()

## grafik AOC
plt.figure()
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - ' + name_classifiers)
plt.legend(loc="lower right")
plt.show()
```

Dari pengolahan data untuk model KNN didapatkan nilai

1. **Confusion Matrik**



Gambar 1.10 Tabel Confusion Matrik

Interpretasi dari Confusion matrik dibawah ini:

1) Kelas 0 (Predicitif Negatif dan Actual Negatif) artinya Terdapat 84 penumpang yang diprediksi dengan benar sebagai kelas 0 (penumpang tidak selamat (Not survive)), dan (Predictif Negatif dan Actual Positif) artinya 21 penumpang yang salah diprediksi sebagai kelas 0 (Not survive Passanger/tidak selamat) padahal actual nya selamat.
2) Kelas 1 (Predictif Positif dan Actual negatif): Ada 45 penumpang yang diprediksi dengan salah sebagai kelas 0 (penumpang selamat) padahal actualnya tidak selamat, dan 28 penumpang yang benar diprediksi kelas 1 (penumpang selamat) dan actual nya selamat.

2. **F1 score: 0.6117573713816116**

   Nilai F1 Score adalah rata-rata harmonik dari precision dan recall. F1 score mencapai nilai terbaik pada 1 dan terburuk pada 0.

   Dilihat dari hasil output data sebesar 0.61 ini bisa diartikan bahwa nilai rata-rata harmonik dari precision dan recall cukup baik

3. **Precision: 0.6184628018963008**

   Nilai precision atau presisi adalah proporsi dari hasil positif yang diidentifikasi dengan benar oleh model dari semua hasil positif yang diprediksi oleh model. Semakin tinggi nilainya, semakin sedikit hasil positif palsu yang dihasilkan oleh model.

   Dari nilai yang didapat bisa diinterpretasikan hampir 61 % hasil positif yang dihasilkan dari data

4. **Recall: 0.6292134831460674**

   Recall adalah proporsi dari instance positif yang berhasil ditemukan dari semua instance positif yang aktual. Nilai yang tinggi menunjukkan bahwa model memiliki kemampuan yang baik dalam menemukan instance positif.

   Dari nilai yang didapat bisa diinterpretasikan hampir 62 % (0.62) data menunjukkan bahwa model memiliki kemampuan yang baik.

5. **Specificity: 0.6292134831460674**

   Nilai Specificity Ini adalah proporsi dari instance negatif yang diprediksi dengan benar dari semua instance negatif yang sebenarnya. Nilai yang tinggi menunjukkan bahwa model jarang membuat kesalahan dalam memprediksi instance positif sebagai negatif. Dari nilai yang didapat bisa diinterpretasikan hampir 62% (0.62) data model jarang membuat kesalahan dalam memprediksi instance positif sebagai negatif.

6. **Logloss: 0.633691355296636**

   Logloss Ini adalah pengukuran tingkat ketidakpastian dari prediksi model, di mana nilai yang rendah menunjukkan bahwa model memiliki tingkat kepercayaan yang tinggi dalam prediksi yang dibuatnya.

   Dari nilai yang didapat bisa diinterpretasikan data hanya memiliki 0.63 tingkat kepercayaan yang tidak begitu tinggi dalam prediksi yang dibuatnya.

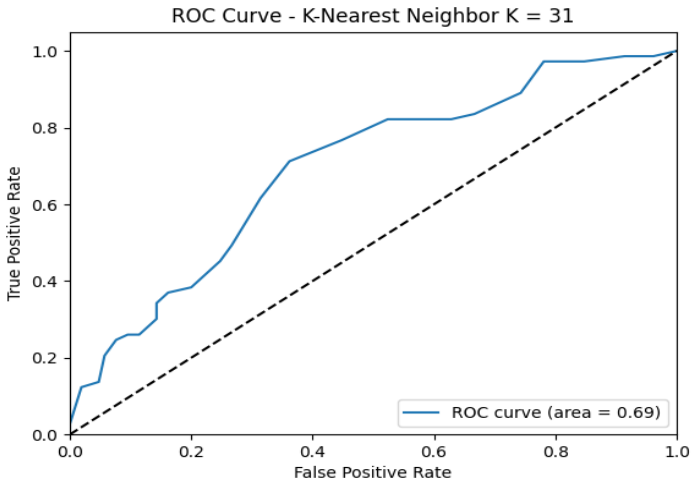
7. **Mcc (Matthews Corrcoef): 0.20213667736336882**

   MCC adalah pengukuran korelasi antara prediksi dan nilai observasi sebenarnya dari sebuah klasifikasi biner. Nilai mendekati 1 menunjukkan korelasi yang sempurna antara prediksi dan nilai sebenarnya.

   Dari nilai 0.20 artinya nilai jauh dari nilai 1, bisa diinterpretasikan menunjukkan korelasi yang tidak bagus antara prediksi dan nilai sebenarnya.

8. **Auroc: 0.694781474233529**

   ROC Curve adalah kurva yang memplot tingkat sensitivitas model terhadap tingkat 1-spesifisitasnya. Area di bawah kurva ROC (Area Under the ROC Curve atau AUROC) juga sering digunakan sebagai metrik evaluasi, di mana nilai AUROC mendekati 1 menunjukkan kinerja yang lebih baik.



Dari nilai ROC 0.69 artinya nilai tidak terlalu mendekati 1 bisa diinterpretasikan kinerja yang cukup baik. Dan dapat dilihat dari grafik pun cenderung mendekati 1.