Nama : Royyan Hali Satrian

NIM : A11.2021.13276

MK : Data Mining 4602

Klasifikasi Obat menggunakan Algoritma

" D-Tree, KNN, GaussianNB, dan SVC "

Atribut :

1. Age : Umur pasien
2. Sex : Jenis Kelamin Pasien
3. Blood Pressure Levels (BP) : Tingkat Tekanan Darah
4. Cholesterol Levels : Tingkat Kolesterol
5. Na to Potassium Ratio : Rasio Natrium Ke Kalium dalam Darah

Jumlah Data :

Terdapat 200 data pada Dataset Drug Classification

Sumber Data :

https://www.kaggle.com/datasets/prathamtripathi/drug-classification

Tahapan Eksperimen :

1. Pemilihan Dataset
2. Visualisasi (Data Grafik)
3. Data Encoding
4. Split Data
5. Modeling
6. Conclusion

Tujuan :

Mengidentifikasi jenis drug (obat-obatan yang di konumsi berdasarkan jenis kelamin, tekanan darah, kolesterol dan kadar kalium ke natrium dalam darah yang merupakan efek dari penggunaan obat-obatan tertentu.

Kesimpulan :

Berdasarkan hasil evaluasi menggunakan confusion matrix didapatkan model machine learning yang memiliki kinerja baik adalah Decisision Tree dengan nilai akurasi, presisi, recall, dan f1-score sempurna

Link Github :

https://github.com/masroyy18/Data-Mining/tree/main/TUGAS%20AKHIR%20DATA%20MINING

Tahapan Ekperimen

1. Pemilihan Dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
#import classification modules
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
```

Import Dataset

```
[2] dataset = pd.read_csv('drug200.csv')
```

```
dataset.head()
```

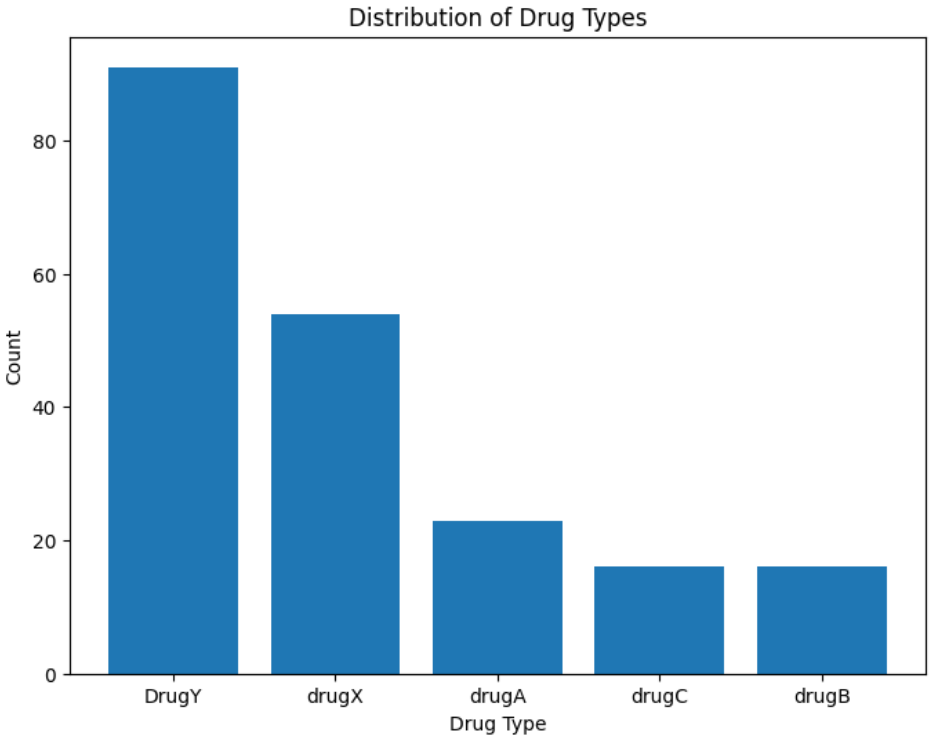
	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	DrugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	DrugY

2. Visualisasi Data (Grafik)

Visualisasi

```
# Count the occurrences of each drug type
drug_counts = dataset['Drug'].value_counts()

# Create a bar plot
plt.figure(figsize=(8, 6))
plt.bar(drug_counts.index, drug_counts.values)
plt.xlabel('Drug Type')
plt.ylabel('Count')
plt.title('Distribution of Drug Types')
plt.show()
```



3. Data Encoding (untuk merubah data menjadi 0,1,2)

```
[15] from sklearn.preprocessing import LabelEncoder

     # Create an instance of LabelEncoder
     label_encoder = LabelEncoder()

     # Encode categorical variables
     dataset['Sex'] = label_encoder.fit_transform(dataset['Sex'])
     dataset['BP'] = label_encoder.fit_transform(dataset['BP'])
     dataset['Cholesterol'] = label_encoder.fit_transform(dataset['Cholesterol'])
     dataset['Drug'] = label_encoder.fit_transform(dataset['Drug'])
```

4. Split Data

```
from sklearn.model_selection import train_test_split

# Split the data into features (X) and target variable (y)
X = dataset.drop('Drug', axis=1)  # Features (all columns except 'Drug')
y = dataset['Drug']  # Target variable ('Drug')

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

5. Modeling

a. DecisionTree

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, f1_score, precision_score, recall_score

dt_classifier = DecisionTreeClassifier()
dt_classifier.fit(X_train, y_train)
y_pred_dt = dt_classifier.predict(X_test)
cm_dt = confusion_matrix(y_test, y_pred_dt)
print("Confusion Matrix:")
print(cm_dt)

#f1 score
f1_dt = f1_score(y_test, y_pred_dt, average='weighted')
print("F1 Score:", f1_dt)

#precision score
precision_dt = precision_score(y_test, y_pred_dt, average='weighted')
print("Precision Score:", precision_dt)

#recall score
recall_dt = recall_score(y_test, y_pred_dt, average='weighted')
print("Recall Score:", recall_dt)
```

```
Confusion Matrix:
[[15  0  0  0  0]
 [ 0  6  0  0  0]
 [ 0  0  3  0  0]
 [ 0  0  0  5  0]
 [ 0  0  0  0 11]]
F1 Score: 1.0
Precision Score: 1.0
Recall Score: 1.0
```

b. KNN

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, f1_score, precision_score, recall_score

knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

#confusion matric
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)

#F1 score
f1 = f1_score(y_test, y_pred, average='weighted')
print("F1 Score:", f1)

#precision score
precision = precision_score(y_test, y_pred, average='weighted')
print("Precision Score:", precision)
```

```
# recall score
recall = recall_score(y_test, y_pred, average='weighted')
print("Recall Score:", recall)
```

```
Accuracy: 0.775
Confusion Matrix:
[[15  0  0  0  0]
 [ 0  5  0  0  1]
 [ 0  0  2  0  1]
 [ 0  3  0  1  1]
 [ 0  1  2  0  8]]
F1 Score: 0.7595238095238096
Precision Score: 0.8208333333333332
Recall Score: 0.775
```

c. GaussianNB

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, f1_score, precision_score, recall_score


nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)
y_pred_nb = nb_classifier.predict(X_test)

# confusion matrix
cm_nb = confusion_matrix(y_test, y_pred_nb)
print("Confusion Matrix:")
print(cm_nb)

# F1 score
f1_nb = f1_score(y_test, y_pred_nb, average='weighted')
print("F1 Score:", f1_nb)

# precision score
precision_nb = precision_score(y_test, y_pred_nb, average='weighted')
print("Precision Score:", precision_nb)
```

```
# recall score
recall_nb = recall_score(y_test, y_pred_nb, average='weighted')
print("Recall Score:", recall_nb)
```

```
Confusion Matrix:
[[12  1  1  1  0]
 [ 0  6  0  0  0]
 [ 0  0  3  0  0]
 [ 0  0  0  5  0]
 [ 0  0  0  0 11]]
F1 Score: 0.9247169497169498
Precision Score: 0.9389880952380952
Recall Score: 0.925
```

d. SVC

```
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, f1_score, precision_score, recall_score


svc_classifier = SVC()
svc_classifier.fit(X_train, y_train)
y_pred_svc = svc_classifier.predict(X_test)

# confusion matrix
cm_svc = confusion_matrix(y_test, y_pred_svc)
print("Confusion Matrix:")
print(cm_svc)

# F1 score
f1_svc = f1_score(y_test, y_pred_svc, average='weighted')
print("F1 Score:", f1_svc)

# precision score
precision_svc = precision_score(y_test, y_pred_svc, average='weighted',zero_division=1)
print("Precision Score:", precision_svc)
```

```
# recall score
recall_svc = recall_score(y_test, y_pred_svc, average='weighted')
print("Recall Score:", recall_svc)
```

```
Confusion Matrix:
[[15  0  0  0  0]
 [ 0  0  0  0  6]
 [ 0  0  0  0  3]
 [ 1  0  0  0  4]
 [ 1  0  0  0 10]]
F1 Score: 0.5133272058823529
Precision Score: 0.8004475703324807
Recall Score: 0.625
```

6. Conclusion

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score

# Initialize the classifiers
dt_classifier = DecisionTreeClassifier()
knn_classifier = KNeighborsClassifier()
nb_classifier = GaussianNB()
svc_classifier = SVC()

classifiers = [dt_classifier, knn_classifier, nb_classifier, svc_classifier]
classifier_names = ['Decision Tree', 'KNN', 'Gaussian Naive Bayes', 'Support Vector Classifier']

# Lists to store evaluation metrics
accuracy_scores = []
f1_scores = []
precision_scores = []
recall_scores = []
```

```
# Evaluate each classifier and collect performance metrics
for classifier in classifiers:
    classifier.fit(X_train, y_train)
    y_pred = classifier.predict(X_test)

    accuracy = 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)
    recall = recall_score(y_test, y_pred, average='weighted')

    accuracy_scores.append(accuracy)
    f1_scores.append(f1)
    precision_scores.append(precision)
    recall_scores.append(recall)
```

```python
# Plotting the performance metrics
x = np.arange(len(classifier_names))
width = 0.2

plt.figure(figsize=(10, 6))
plt.bar(x, accuracy_scores, width, label='Accuracy')
plt.bar(x + width, f1_scores, width, label='F1 Score')
plt.bar(x + (2 * width), precision_scores, width, label='Precision')
plt.bar(x + (3 * width), recall_scores, width, label='Recall')

plt.xlabel('Classifier')
plt.ylabel('Score')
plt.title('Performance Evaluation of Classifiers')
plt.xticks(x + width * 1.5, classifier_names, rotation=45)
plt.legend()
plt.tight_layout()
plt.show()
```