21162101012_CBA_Yash_Lakhtariya

ML Practical 6

Aim - Design following model for given dataset and find the best suitable model using confusion matrix, f-score, accuracy, recall and precision matrices.

- decision tree
- logistic regression
- KNN

```
[28] import pandas as pd
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import LabelEncoder, StandardScaler
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.linear_model import LogisticRegression
     from sklearn.neighbors import KNeighborsClassifier
     from sklearn.metrics import confusion_matrix, accuracy_score, f1_score, recall_score, precision_score
```

```
[29] df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/ML_7/p6_drug200.csv')

     df.columns = ['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug']
```

```
     df.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

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
le_sex = LabelEncoder()
le_bp = LabelEncoder()
le_chol = LabelEncoder()
le_drug = LabelEncoder()
```

```
df['Sex'] = le_sex.fit_transform(df['Sex'])
df['BP'] = le_bp.fit_transform(df['BP'])
df['Cholesterol'] = le_chol.fit_transform(df['Cholesterol'])
df['Drug'] = le_drug.fit_transform(df['Drug'])
```

```
# Split the data into features (X) and target (y)
X = df[['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K']]
y = df['Drug']

# Train-test split (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
scaler = StandardScaler()
X = scaler.fit_transform(X)

models = {
    'Decision Tree': DecisionTreeClassifier(),
    'Logistic Regression': LogisticRegression(max_iter=10000),  # Increase max_iter to 1000
    'KNN': KNeighborsClassifier(n_neighbors=3)
}
```

```
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    cm = confusion_matrix(y_test, y_pred)
    accuracy = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred, average='weighted')
    recall = recall_score(y_test, y_pred, average='weighted')
    precision = precision_score(y_test, y_pred, average='weighted')

    print(f'\nConfusion Matrix:\n{cm}')
    print(f'\n\tAccuracy: {accuracy:.2f}')
    print(f'\tF1-Score: {f1:.2f}')
    print(f'\tRecall: {recall:.2f}')
    print(f'\tPrecision: {precision:.2f}')
    print()
    print('-' * 30)
```

```
for name, model in models.items():
    print(f'\nEvaluating {name}...')
    model.fit(X_train, y_train)
    evaluate_model(model, X_test, y_test)
```

```

Evaluating Decision Tree...

Confusion Matrix:
[[ 6  0  0  0  0]
 [ 0  3  0  0  0]
 [ 0  0  5  0  0]
 [ 0  0  0 11  0]
 [ 0  0  0  0 15]]

        Accuracy: 1.00
        F1-Score: 1.00
        Recall: 1.00
        Precision: 1.00

------------------------------

Evaluating Logistic Regression...

Confusion Matrix:
[[ 6  0  0  0  0]
 [ 0  3  0  0  0]
 [ 0  0  2  3  0]
 [ 0  1  0 10  0]
 [ 0  0  0  0 15]]

        Accuracy: 0.90
        F1-Score: 0.89
        Recall: 0.90
        Precision: 0.92

------------------------------

Evaluating KNN...

Confusion Matrix:
[[ 5  0  0  1  0]
 [ 0  2  0  1  0]
 [ 3  0  1  1  0]
 [ 1  2  0  8  0]
 [ 0  0  0  0 15]]

        Accuracy: 0.78
        F1-Score: 0.76
        Recall: 0.78
        Precision: 0.82

    ------------------------------
```

--> Hence, for the given dataset, the perfect model is **Decision Tree** with full **accuracy of 100%** and the least suitable one is **KNN** with **accuracy 78%**