mnttxydyp

September 10, 2024

[5]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score

# Step 1: Load the dataset
file_path = 'vgsales.csv'
df = pd.read_csv(file_path)

# Step 2: Explore the dataset
print(df.head())
print(df.info())

# Drop any rows with missing values (for simplicity)
df = df.dropna()

# For this exercise, we need to create a classification target
# Example: If the 'Global_Sales' column exists, let's create a target:
# High sales (1) vs Low sales (0) using a threshold (e.g., median)

# Check if 'Global_Sales' column exists
if 'Global_Sales' in df.columns:
    df['High_Sales'] = (df['Global_Sales'] > df['Global_Sales'].median()).
 ↪astype(int)
    X = df.drop(['Global_Sales', 'High_Sales'], axis=1)  # Features
    y = df['High_Sales']  # Target
else:
    # Modify this section based on available columns if 'Global_Sales' doesn't␣
 ↪exist
    print("Modify the code to choose a suitable target column for␣
 ↪classification.")

# For simplicity, let's use only numerical columns for classification
X = df.select_dtypes(include=['float64', 'int64'])  # Use numeric features
```

```
y = df['High_Sales']  # Our target variable

# Step 3: Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

# Step 4: Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Step 5: Model Development

# 5.1 Logistic Regression
lr = LogisticRegression(max_iter=200)
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)

# 5.2 k-Nearest Neighbors (k-NN)
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)

# 5.3 Decision Tree
dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)
y_pred_dt = dt.predict(X_test)

# Step 6: Model Evaluation

# Logistic Regression Metrics
print("Logistic Regression Classification Report:\n")
print(classification_report(y_test, y_pred_lr))

# k-NN Metrics
print("k-NN Classification Report:\n")
print(classification_report(y_test, y_pred_knn))

# Decision Tree Metrics
print("Decision Tree Classification Report:\n")
print(classification_report(y_test, y_pred_dt))

# Compare Accuracy
accuracy_lr = accuracy_score(y_test, y_pred_lr)
accuracy_knn = accuracy_score(y_test, y_pred_knn)
accuracy_dt = accuracy_score(y_test, y_pred_dt)
```

```
print(f"Logistic Regression Accuracy: {accuracy_lr:.4f}")
print(f"k-NN Accuracy: {accuracy_knn:.4f}")
print(f"Decision Tree Accuracy: {accuracy_dt:.4f}")
```

```
   Rank                      Name Platform    Year         Genre Publisher  \
0     1                Wii Sports      Wii  2006.0        Sports  Nintendo
1     2          Super Mario Bros.      NES  1985.0      Platform  Nintendo
2     3             Mario Kart Wii      Wii  2008.0        Racing  Nintendo
3     4          Wii Sports Resort      Wii  2009.0        Sports  Nintendo
4     5  Pokemon Red/Pokemon Blue       GB  1996.0  Role-Playing  Nintendo

   NA_Sales  EU_Sales  JP_Sales  Other_Sales  Global_Sales
0     41.49     29.02      3.77         8.46         82.74
1     29.08      3.58      6.81         0.77         40.24
2     15.85     12.88      3.79         3.31         35.82
3     15.75     11.01      3.28         2.96         33.00
4     11.27      8.89     10.22         1.00         31.37
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16598 entries, 0 to 16597
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Rank          16598 non-null  int64
 1   Name          16598 non-null  object
 2   Platform      16598 non-null  object
 3   Year          16327 non-null  float64
 4   Genre         16598 non-null  object
 5   Publisher     16540 non-null  object
 6   NA_Sales      16598 non-null  float64
 7   EU_Sales      16598 non-null  float64
 8   JP_Sales      16598 non-null  float64
 9   Other_Sales   16598 non-null  float64
 10  Global_Sales  16598 non-null  float64
dtypes: float64(6), int64(1), object(4)
memory usage: 1.1+ MB
None
Logistic Regression Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2472
           1       1.00      1.00      1.00      2416

    accuracy                           1.00      4888
   macro avg       1.00      1.00      1.00      4888
weighted avg       1.00      1.00      1.00      4888
```

```
k-NN Classification Report:

              precision    recall  f1-score   support

           0       0.99      0.99      0.99      2472
           1       0.99      0.99      0.99      2416

    accuracy                           0.99      4888
   macro avg       0.99      0.99      0.99      4888
weighted avg       0.99      0.99      0.99      4888

Decision Tree Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2472
           1       1.00      1.00      1.00      2416

    accuracy                           1.00      4888
   macro avg       1.00      1.00      1.00      4888
weighted avg       1.00      1.00      1.00      4888

Logistic Regression Accuracy: 0.9982
k-NN Accuracy: 0.9943
Decision Tree Accuracy: 1.0000
```

[ ]: