ARTIFICIAL INTELLIGENCE

PROJECT : VIDEO GAME SALES

GROUP MEMBERS:

M.IBRAHIM CHATTHA

M.SUBHAN ALI AWAN

ASAD ZULFIQAR

vgsales (1).csv

DATASET

Code Description

The code conducts a regression and classification analysis to predict and classify the global video game sales using regional sales data. The dataset, vgsales.csv, is the data set on video game sales, and the aim is to predict the global sales of video games using the sales data from Japan, Europe, and other regions.

Steps and Operations:

Data Loading:

The code starts by importing all the required libraries: pandas, numpy, matplotlib, sklearn models, and metrics.

It reads the dataset vgsales.csv using pandas.read_csv() and displays the first few rows using.head().

Data Preparation:

The feature set X is created by selecting columns for sales data from Japan (JP_Sales), Europe (EU_Sales), and Other regions (Other_Sales).

The target variable y is set as the Global_Sales.

The data is split into training and testing sets using train_test_split() from sklearn.model_selection with 80% of data for training and 20% for testing.

Linear Regression Model:

A Linear Regression model is instantiated and trained using X_train and y_train with.fit().

Predictions for X_test are made using.predict(), and the mean squared error (MSE) is calculated to assess model performance using mean_squared_error().

The coefficients and intercept of the linear regression model are printed.

A scatter plot is created for actual vs. predicted global sales. The best-fit line is plotted for reference.

Random Forest Regressor Model:

A Random Forest Regressor model is created with 100 trees and trained on the same training data.

Prediction is done, and MSE is calculated for the Random Forest model.

Similar to the linear regression, a scatter plot is created to show actual vs. predicted sales.

Model Evaluation and Visualization:

Both models (Linear Regression and Random Forest) are visually compared side-by-side using a 2x2 grid layout.
A red dashed line is added to indicate the ideal predictions (where actual equals predicted).

Categorizing Sales:

A function categorize_sales() is defined to categorize the sales value into two classes:
'Low' for sales less than 5 million.
'High' for sales greater than or equal to 5 million.
This function creates a true categorical sales labels (y_test) and the predicted sales labels from both models (lr_y_pred and rf_y_pred).

Confusion Matrix Calculation:

The confusion matrixes are calculated by using confusion_matrix() from sklearn.metrics for both the models: Linear Regression, and Random Forest. The matrices reflect how good or bad these models are in a classification of sales as 'Low' or 'High'.
The accuracy of each model is calculated by finding correct predictions over total predictions.

Final Output:

The confusion matrices and accuracy values for both models are printed in output, which provides an insight into the classification performance.

Algorithm Used:

Linear Regression Algorithm:

Linear regression models the relationship between input features (JP_Sales, EU_Sales, Other_Sales) and the target variable

(Global_Sales) by fitting a linear equation. It minimizes the squared differences between actual and predicted values using the least squares method.

Random Forest Regressor Algorithm:

The Random Forest Regressor constructs multiple decision trees, defaults here are set to 100. It trains each of the trees on randomly selecting data. It combines all the trees' predictions together with averaging in making a final prediction, which minimizes overfitting and improves accuracy.

Classification (Prediction)

After regression predictions, using the categorize_sales() function, the sales values are categorized into 'Low' and 'High' categories based on a threshold of 5 million.

Confusion Matrix and Calculation of Accuracy:

This confusion matrix compares the predictions with the proper categories (Low or High). The formula given is used to calculate the accuracy.

CODE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
```

```
data = pd.read_csv("vgsales.csv")
data.head()
X = data[['JP_Sales', 'EU_Sales', 'Other_Sales']]
y = data['Global_Sales']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_y_pred = lr_model.predict(X_test)
lr_mse = mean_squared_error(y_test, lr_y_pred)
print("Linear Regression MSE:", lr_mse)
print("Linear Regression Coefficients:", lr_model.coef_)
print("Linear Regression Intercept:", lr_model.intercept_)
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.scatter(y_test, lr_y_pred, color='blue', alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
color='red', linestyle='--')  # Ideal line
plt.title('Linear Regression: Actual vs Predicted')
plt.xlabel('Actual Global Sales')
plt.ylabel('Predicted Global Sales')
plt.grid(True)
rf_model = RandomForestRegressor(n_estimators=100,
random_state=42)
rf_model.fit(X_train, y_train)
rf_y_pred = rf_model.predict(X_test)
rf_mse = mean_squared_error(y_test, rf_y_pred)
print("\nRandom Forest Regressor MSE:", rf_mse)
plt.subplot(1, 2, 2)
plt.scatter(y_test, rf_y_pred, color='green', alpha=0.6)
```

```
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
color='red', linestyle='--')  # Ideal line
plt.title('Random Forest Regressor: Actual vs Predicted')
plt.xlabel('Actual Global Sales')
plt.ylabel('Predicted Global Sales')
plt.grid(True)

# Show the plots
plt.tight_layout()
plt.show()
plt.subplot(1, 2, 2)
plt.scatter(y_test, rf_y_pred, color='green', alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
color='red', linestyle='--')  # Ideal line
plt.title('Random Forest Regressor: Actual vs Predicted')
plt.xlabel('Actual Global Sales')
plt.ylabel('Predicted Global Sales')
plt.grid(True)
from sklearn.metrics import confusion_matrix
import pandas as pd

def categorize_sales(value):
   if value < 5:
      return 'Low'   # Merging "Low" and "Medium" into 'Low'
   else:
      return 'High'  # Treating "High" sales as 'High'

# Assuming y_test, lr_y_pred, and rf_y_pred are already defined
y_test_cat = y_test.apply(categorize_sales)
lr_y_pred_cat = pd.Series(lr_y_pred).apply(categorize_sales)
```

```
rf_y_pred_cat = pd.Series(rf_y_pred).apply(categorize_sales)

# Compute confusion matrices
lr_conf_matrix = confusion_matrix(y_test_cat, lr_y_pred_cat,
labels=['Low', 'High'])
print("Linear Regression 2x2 Confusion Matrix:\n", lr_conf_matrix)

rf_conf_matrix = confusion_matrix(y_test_cat, rf_y_pred_cat,
labels=['Low', 'High'])
print("\nRandom Forest 2x2 Confusion Matrix:\n", rf_conf_matrix)

# Optionally, print out the accuracy for each model
lr_accuracy = (lr_conf_matrix[0, 0] + lr_conf_matrix[1, 1]) /
lr_conf_matrix.sum()
rf_accuracy = (rf_conf_matrix[0, 0] + rf_conf_matrix[1, 1]) /
rf_conf_matrix.sum()

print("\nLinear Regression Accuracy:", lr_accuracy)
print("Random Forest Accuracy:", rf_accuracy)
```