2q8pk3cqw

February 7, 2025

___ Decision Tree ___

Step 1: Identify Code Sections for the Assignment

Review your Jupyter Notebook: Go through your code and pinpoint sections suitable for the assignment. These could be parts where students need to:

Implement a specific algorithm or function. Apply data preprocessing techniques. Perform model evaluation. Visualize data. Choose sections to remove or modify: Select the code parts you want students to fill in. Consider the assignment's difficulty and learning objectives.

Step 2: Create Instructions and Placeholders

Add Markdown cells for instructions: Insert Markdown cells before each code section you've identified for the assignment.

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

[8]:
```
# ====================================
# Load in our dataset
# ====================================
ic_data = pd.read_csv("IceCreamData.csv")
```

[9]:
```
ic_data.head()
```

[9]:

	Temperature	Revenue
0	24.566884	534.799028
1	26.005191	625.190122
2	27.790554	660.632289
3	20.595335	487.706960
4	11.503498	316.240194

[10]:
```
# Get the data value in separate variables
X = ic_data['Temperature'].values
y = ic_data['Revenue'].values
```

[6]:
```
# ====================================
# Data preprocessing
# ====================================

# Using standard scalar encoding
en_sc = StandardScaler()
X = en_sc.fit_transform(X.reshape(-1,1))
```

[7]:
```
# ====================================
# Split the data into train and test part
# ====================================
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,␣
 ↪random_state = 5)
```

0.0.1 Task 3: Train the Decision Tree Model

In this section, you will train a Decision Tree Regressor model using the training data.

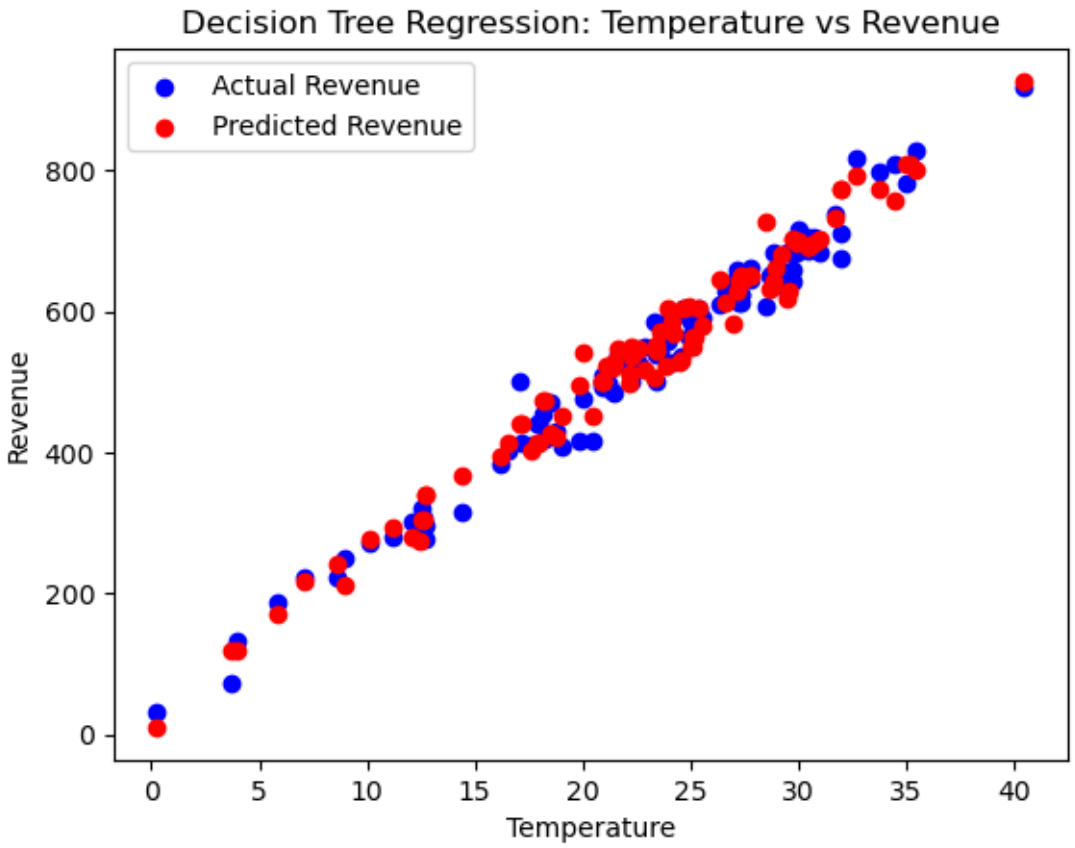
Instructions:

1. Initialize a DecisionTreeRegressor object.
2. Fit the model to the training data (`x_train` and `y_train`).

[15]:
```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Load the dataset
ic_data = pd.read_csv("IceCreamData.csv")

# Define features (X) and target variable (y)
X = ic_data[['Temperature']]  # Feature
y = ic_data['Revenue']  # Target variable

# Split the dataset into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Step 1: Initialize the Decision Tree Regressor
dt_regressor = DecisionTreeRegressor(random_state=42)
```

```
# Step 2: Train the model on the training data
dt_regressor.fit(X_train, y_train)

# Step 3: Make predictions on the test set
y_pred = dt_regressor.predict(X_test)

# Step 4: Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R2 Score:", r2)

# Step 5: Visualize the Decision Tree Predictions
plt.scatter(X_test, y_test, color="blue", label="Actual Revenue")
plt.scatter(X_test, y_pred, color="red", label="Predicted Revenue")
plt.xlabel("Temperature")
plt.ylabel("Revenue")
plt.title("Decision Tree Regression: Temperature vs Revenue")
plt.legend()
plt.show()
```

```
Mean Squared Error: 1328.7245076491977
R2 Score: 0.9534772958686918
```



[18]:
```
# Predict the model with testing data
labels = dt_regressor.predict(X_test)

# Display predictions
print("Predicted Revenue:", labels)
```

```
Predicted Revenue: [702.9940111 649.729072  603.2329422 521.7754452 612.2437215
278.4182651
 293.9263927 303.7343815 528.1162401 696.6401775 733.215828  414.423028
 413.9140669 679.3177906  10.        216.183462  550.2785159 569.6187562
 563.2509867 702.9940111 523.1245467 756.9625616 726.2337713 526.5470649
 926.0671533 662.5589903 303.7343815 642.3498137 773.9247547 690.7892959
 473.5681122 496.0112948 702.6236136 809.6720534 651.5043041 499.4583433
 793.079011  339.1095829 276.3733742 118.8121496 550.7014036 241.2785475
 553.1196514 395.2737497 473.5681122 579.3073878 531.7424848 118.8121496
 541.2936627 546.6938576 581.0740052 520.4703098 451.4507843 521.7754452
 570.9909316 642.3498137 603.3053386 629.8937918 170.2377561 631.3182368
 550.7014036 696.7166402 516.5486011 642.3498137 618.2357655 441.5087331
 545.9039291 499.4583433 773.9247547 274.0656189 339.1095829 450.4732071
 807.5412872 401.4330183 512.5881071 698.9718063 800.2024937 421.621505
```

```
 499.4583433 499.4583433 506.4321353 496.4613625 641.0253891 542.8391063
 603.3246306 690.7892959 212.5917401 627.6508336 563.2509867 774.1080813
 413.9140669 535.8667293 646.2669458 367.9407438 586.150568  607.5421478
 651.5043041 633.5040087 441.5087331 427.2113597]
```

[19]:
```python
# ====================
# Evaluate the model
# ====================
#from sklearn.metrics import mean_squared_error, r2_score

# ====================
# Evaluate the model
# ====================

# Mean Squared Error (MSE)
mse = mean_squared_error(y_test, labels)

# Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)

# R² Score
r2 = r2_score(y_test, labels)

# Print evaluation metrics
print("Mean Squared Error (MSE):", mse)
print("Root Mean Squared Error (RMSE):", rmse)
print("R² Score:", r2)
```

```
Mean Squared Error (MSE): 1328.7245076491977
Root Mean Squared Error (RMSE): 36.451673591883235
R² Score: 0.9534772958686918
```