```
# 1. Import libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import BaggingRegressor, RandomForestRegressor
from sklearn.metrics import mean_squared_error
from xgboost import XGBRegressor
import matplotlib.pyplot as plt

# 2. Load and preprocess data
df = pd.read_csv('amazon_sales_data-2025.csv')

# Profit calculation
df['Estimated_Cost'] = 0.7 * df['Price'] * df['Quantity']
df['Profit'] = df['Total Sales'] - df['Estimated_Cost']

# Drop non-predictive columns
df = df.drop(['Order ID', 'Date', 'Customer Name'], axis=1)

# Encode categorical variables
categorical_cols = ['Product', 'Category', 'Customer Location', 'Payment Method',
'Status']
df = pd.get_dummies(df, columns=categorical_cols, drop_first=True)

# 3. Train-test split
X = df.drop(['Profit', 'Estimated_Cost', 'Total Sales'], axis=1)
y = df['Profit']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# 4. Baseline: Decision Tree
tree = DecisionTreeRegressor(max_depth=4, random_state=42)
tree.fit(X_train, y_train)
y_pred_tree = tree.predict(X_test)
rmse_tree = np.sqrt(mean_squared_error(y_test, y_pred_tree))
print("Decision Tree RMSE:", round(rmse_tree, 4))

# 5. Bagging
bagging = BaggingRegressor(n_estimators=100, random_state=42)
bagging.fit(X_train, y_train)
y_pred_bag = bagging.predict(X_test)
rmse_bag = np.sqrt(mean_squared_error(y_test, y_pred_bag))
print("Bagging RMSE:", round(rmse_bag, 4))

# 6. Random Forest
rf = RandomForestRegressor(n_estimators=100, max_features='sqrt', random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
rmse_rf = np.sqrt(mean_squared_error(y_test, y_pred_rf))
print("Random Forest RMSE:", round(rmse_rf, 4))

# 7. Boosting (XGBoost)
xgb = XGBRegressor(n_estimators=200, learning_rate=0.1, max_depth=3,
random_state=42)
xgb.fit(X_train, y_train)
y_pred_xgb = xgb.predict(X_test)
rmse_xgb = np.sqrt(mean_squared_error(y_test, y_pred_xgb))
print("XGBoost RMSE:", round(rmse_xgb, 4))
```

```
# 8. BART (using bartpy)
try:
    from bartpy.sklearnmodel import SklearnModel
    bart = SklearnModel()
    bart.fit(X_train.values, y_train.values)
    y_pred_bart = bart.predict(X_test.values)
    rmse_bart = np.sqrt(mean_squared_error(y_test, y_pred_bart))
    print("BART RMSE:", round(rmse_bart, 4))
except ImportError:
    print("BART (bartpy) is not installed. Run: pip install bartpy")
    rmse_bart = None

# 9. Variable Importance Plots

features = X.columns

# Random Forest Importance
importances_rf = rf.feature_importances_
indices_rf = np.argsort(importances_rf)[::-1]
plt.figure(figsize=(10,6))
plt.title("Feature Importances (Random Forest)")
plt.bar(range(X.shape[1]), importances_rf[indices_rf])
plt.xticks(range(X.shape[1]), features[indices_rf], rotation=90)
plt.tight_layout()
plt.show()

# XGBoost Importance
importances_xgb = xgb.feature_importances_
indices_xgb = np.argsort(importances_xgb)[::-1]
plt.figure(figsize=(10,6))
plt.title("Feature Importances (XGBoost)")
plt.bar(range(X.shape[1]), importances_xgb[indices_xgb])
plt.xticks(range(X.shape[1]), features[indices_xgb], rotation=90)
plt.tight_layout()
plt.show()

# BART Importance (Permutation Importance)
if rmse_bart is not None:
    from sklearn.inspection import permutation_importance
    result = permutation_importance(bart, X_test.values, y_test.values,
n_repeats=10, random_state=42)
    importances_bart = result.importances_mean
    indices_bart = np.argsort(importances_bart)[::-1]
    plt.figure(figsize=(10,6))
    plt.title("Feature Importances (BART via Permutation)")
    plt.bar(range(X.shape[1]), importances_bart[indices_bart])
    plt.xticks(range(X.shape[1]), features[indices_bart], rotation=90)
    plt.tight_layout()
    plt.show()

# 10. Tuning Curve Example: Random Forest
rmse_list = []
n_trees_list = [10, 50, 100, 200, 300]
for n in n_trees_list:
    rf_tune = RandomForestRegressor(n_estimators=n, random_state=42)
    rf_tune.fit(X_train, y_train)
    y_pred_tune = rf_tune.predict(X_test)
    rmse_list.append(np.sqrt(mean_squared_error(y_test, y_pred_tune)))
```

```
plt.plot(n_trees_list, rmse_list, marker='o')
plt.xlabel('Number of Trees')
plt.ylabel('Test RMSE')
plt.title('Random Forest: RMSE vs. Number of Trees')
plt.show()

# 11. Print Summary Table
print("\nSummary Table (Test RMSE):")
print(f"Decision Tree: {round(rmse_tree,2)}")
print(f"Bagging: {round(rmse_bag,2)}")
print(f"Random Forest: {round(rmse_rf,2)}")
print(f"XGBoost: {round(rmse_xgb,2)}")
if rmse_bart is not None:
    print(f"BART: {round(rmse_bart,2)}")
else:
    print("BART: Not available (bartpy not installed)")
```