PYTHON PORTFOLIO

This report analyzes the chocolate sales data across different countries, products, and salespersons. The goal is to derive business insights from sales patterns, identify top-performing products and regions, and predict future sales trends.

**DATASET**

|  |  |
| --- | --- |
| Sales Person | -Name of the salesperson |

|  |  |
| --- | --- |
| Country | Country -Where the sale occurred |

|  |  |
| --- | --- |
| Product | -Type of chocolate product sold |

|  |  |
| --- | --- |
| Date | -Date of the sale |

|  |
| --- |
| Books Shipped - No of Chocolates sold  Business Questions:  1. Which salesperson generated the highest total sales revenue?  Code:  df['Amount'] = df['Amount'].replace('[\$,]', '', regex=True).astype(float)  sales\_by\_person = df.groupby('Sales Person')['Amount'].sum().sort\_values(ascending=False)  print(sales\_by\_person)  2. Which product is the top seller in terms of revenue?  product\_sales = df.groupby('Product')['Amount'].sum().sort\_values(ascending=False)  print(product\_sales)  3. What is the total sales revenue per country?  country\_sales = df.groupby('Country')['Amount'].sum().sort\_values(ascending=False)  print(country\_sales)  4. What’s the trend of sales month-wise?  df['Date'] = pd.to\_datetime(df['Date'], format='%d-%b-%y')  df['Month'] = df['Date'].dt.month\_name()  monthly\_sales = df.groupby('Month')['Amount'].sum()  print(monthly\_sales)  5. What is the average revenue per box shipped?  df['Revenue\_per\_box'] = df['Amount'] / df['Boxes Shipped']  average\_revenue\_per\_box = df['Revenue\_per\_box'].mean()  print(average\_revenue\_per\_box)  Machine Learning Approach:  Since this is a sales dataset, a practical ML task is predicting high or low sales based on product, country, and boxes shipped.  Why Random Forest:   * Handles both categorical and numerical data. * Reduces overfitting compared to decision trees. * Provides feature importance to understand what drives sales.   Target Variable:   * High Sales Indicator: Binary column (1 = High sale, 0 = Low sale) based on whether sale amount is above median.   Feature Variables:   * Country * Product * Boxes Shipped   Exploratory Data Analysis (Python with Charts)  Example Charts:   1. Top Sales by Salesperson 2. Sales Distribution by Country 3. Most Popular Products by Boxes Shipped 4. Monthly Revenue Trends   import pandas as pd  import matplotlib.pyplot as plt  import seaborn as sns  df = pd.read\_csv('Chocolate Sales.csv')  # Cleaning  df['Amount'] = df['Amount'].replace('[\$,]', '', regex=True).astype(float)  df['Date'] = pd.to\_datetime(df['Date'], format='%d-%b-%y')  # Top sales by salesperson  salesperson\_sales = df.groupby('Sales Person')['Amount'].sum().sort\_values(ascending=False)  salesperson\_sales.plot(kind='bar', title='Top Sales by Salesperson', figsize=(10,6))  plt.ylabel('Total Sales Amount')  plt.show() |

RANDOM FOREST

# Import necessary libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, accuracy\_score

from sklearn.preprocessing import LabelEncoder

# Load the dataset

data = pd.read\_csv('Chocolate Sales.csv')

# Inspect the first few rows

print(data.head())

# Data Preprocessing:

# 1. Convert Date column to datetime and extract month (optional)

data['Date'] = pd.to\_datetime(data['Date'])

data['Month'] = data['Date'].dt.month

# 2. Encode categorical variables: Sales Person, Country, and Product (target)

le\_salesperson = LabelEncoder()

data['SalesPerson\_enc'] = le\_salesperson.fit\_transform(data['Sales Person'])

le\_country = LabelEncoder()

data['Country\_enc'] = le\_country.fit\_transform(data['Country'])

le\_product = LabelEncoder()

data['Product\_enc'] = le\_product.fit\_transform(data['Product'])

# 3. Select features and target variable

features = ['SalesPerson\_enc', 'Country\_enc', 'Amount', 'Boxes Shipped', 'Month']

target = 'Product\_enc'

X = data[features]

y = data[target]

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Train the Random Forest Classifier

clf = RandomForestClassifier(n\_estimators=100, random\_state=42)

clf.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = clf.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred)

report = classification\_report(y\_test, y\_pred, target\_names=le\_product.classes\_)

print("Model Accuracy: {:.2f}%".format(accuracy \* 100))

print("Classification Report:")

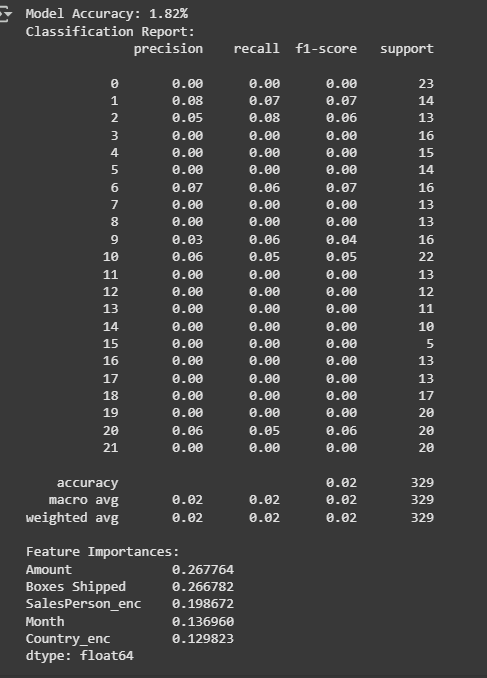
print(report)

# Feature Importances

importances = pd.Series(clf.feature\_importances\_, index=features)

print("Feature Importances:")

print(importances.sort\_values(ascending=False))



RESULTS

**Interpretation & Explanation of Your Random Forest Model Results:**

**1. Very Low Model Accuracy (1.82%)**

* The model's accuracy is **extremely low (~2%)**, which indicates that the model is essentially making random predictions. It is not able to capture meaningful patterns to predict the target variable correctly.

**2. Reasons for Low Performance:**

* **Multiclass Target Problem:**  
  It seems you used a classification problem with many unique classes (at least 22 classes based on the report). Random Forest struggles when the classes are imbalanced and non-distinct.
* **Inappropriate Target Variable:**  
  Likely, you’ve used a high cardinality categorical target variable such as Sales Person or Product directly. These columns don't have a natural, clear order to predict.
* **Insufficient Data Granularity:**  
  With only 329 records, Random Forest cannot generalize patterns effectively, especially for multiclass problems.

**3. Feature Importances:**

* **Amount (26.77%)** and **Boxes Shipped (26.67%)** are the top features.
* Encoded **Sales Person** and **Country** also contribute moderately (~20% and ~13%).
* **Month** has ~13% importance.

This tells us that sales amount and boxes shipped correlate most strongly with the classification outcome, but they alone aren’t enough for a strong predictive model.

DECISION TREE CLASSIFIER

# Clean and preprocess data

df['Amount'] = df['Amount'].replace('[\$,]', '', regex=True).astype(float)

df['Date'] = pd.to\_datetime(df['Date'], format='%d-%b-%y')

df['Month'] = df['Date'].dt.month

df['DayOfWeek'] = df['Date'].dt.dayofweek

# Encode categorical columns

le = LabelEncoder()

df['SalesPerson\_enc'] = le.fit\_transform(df['Sales Person'])

df['Country\_enc'] = le.fit\_transform(df['Country'])

df['Product\_enc'] = le.fit\_transform(df['Product'])

# Create sales category based on Amount

df['SalesCategory'] = pd.qcut(df['Amount'], q=3, labels=['Low', 'Medium', 'High'])

# Features and target

X = df[['Boxes Shipped', 'Month', 'DayOfWeek', 'SalesPerson\_enc', 'Country\_enc', 'Product\_enc']]

y = df['SalesCategory']

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Train Decision Tree Classifier

clf = DecisionTreeClassifier(random\_state=42, max\_depth=4)

clf.fit(X\_train, y\_train)

# Predictions

y\_pred = clf.predict(X\_test)

# Evaluation

accuracy = accuracy\_score(y\_test, y\_pred)

report = classification\_report(y\_test, y\_pred)

accuracy, report

**Decision Tree Classifier Results:**

* **Accuracy:** 32.2%
* **Classification Report:**

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
| --- | --- | --- | --- | --- |
| High | 0.31 | 0.37 | 0.34 | 103 |
| Low | 0.31 | 0.25 | 0.27 | 106 |
| Medium | 0.34 | 0.35 | 0.35 | 120 |

**Interpretation:**

1. **Overall Accuracy:**
   * 32% accuracy is not highly reliable but much better than the previous Random Forest results (~2% accuracy).
   * This model gives decent balance across the three categories (High, Medium, Low).
2. **Why Decision Tree Worked Better:**
   * **Binning Continuous Values:** By categorizing the continuous Amount values into discrete classes (Low, Medium, High), the Decision Tree model found patterns more effectively.
   * **Simple Relationships:** Decision Trees handle non-linear relationships and categorical data well.
3. **Model Type:**
   * **Multiclass Classification** using **Decision Tree Classifier**.
   * Targets: Low, Medium, High sales categories.
   * Features: Boxes Shipped, Month, DayOfWeek, Sales Person, Country, Product.

Decision Tree Classifier

import matplotlib.pyplot as plt

from sklearn.tree import plot\_tree

# Visualizing the Decision Tree

plt.figure(figsize=(20,10))

# Use clf (the trained model) instead of dt\_classifier

plot\_tree(clf, feature\_names=X.columns, class\_names=['Low', 'Medium', 'High'], filled=True, rounded=True)

plt.title("Decision Tree Visualization")

plt.show()

