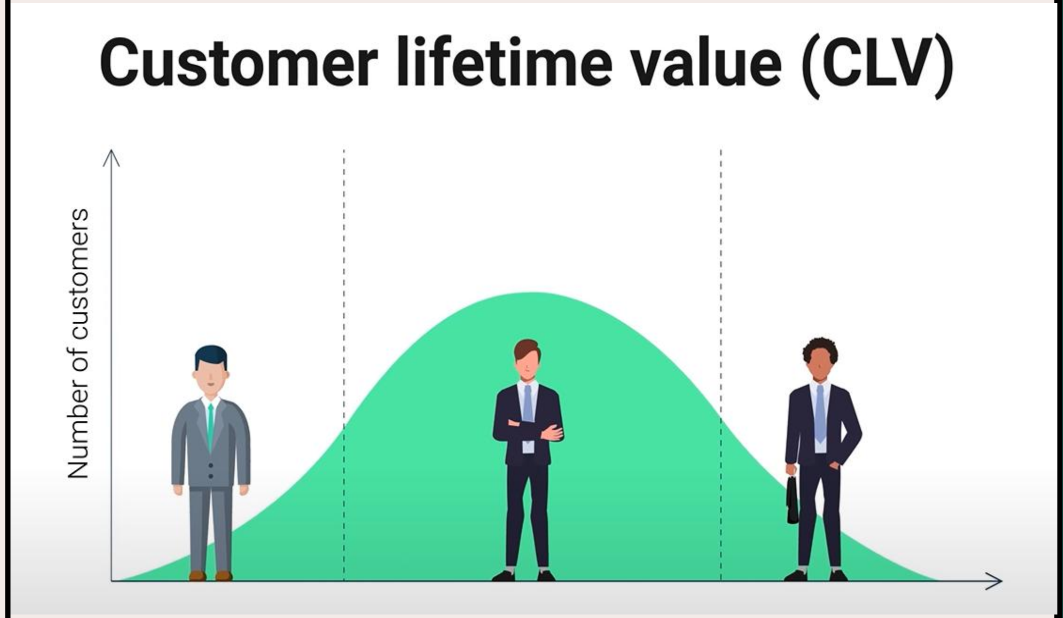
Predicting Customer Lifetime Value (CLV) Using Python

Data-Driven Approach to Estimate Future Customer Spending



Presented By:
2023BCS109 Pruthviraj Jadhav
2023BCS101 Krishna Kalyan

Under the Guidance of:
Prof. Amit Nandedkar Sir

Agenda

1. Introduction
2. Primary Goals
3. Plan
4. Output
5. Summary

What is CLV

<u>Customer Lifetime Value (CLV) is a metric that estimates the total net profit a company can generate from a customer over their entire relationship.</u>

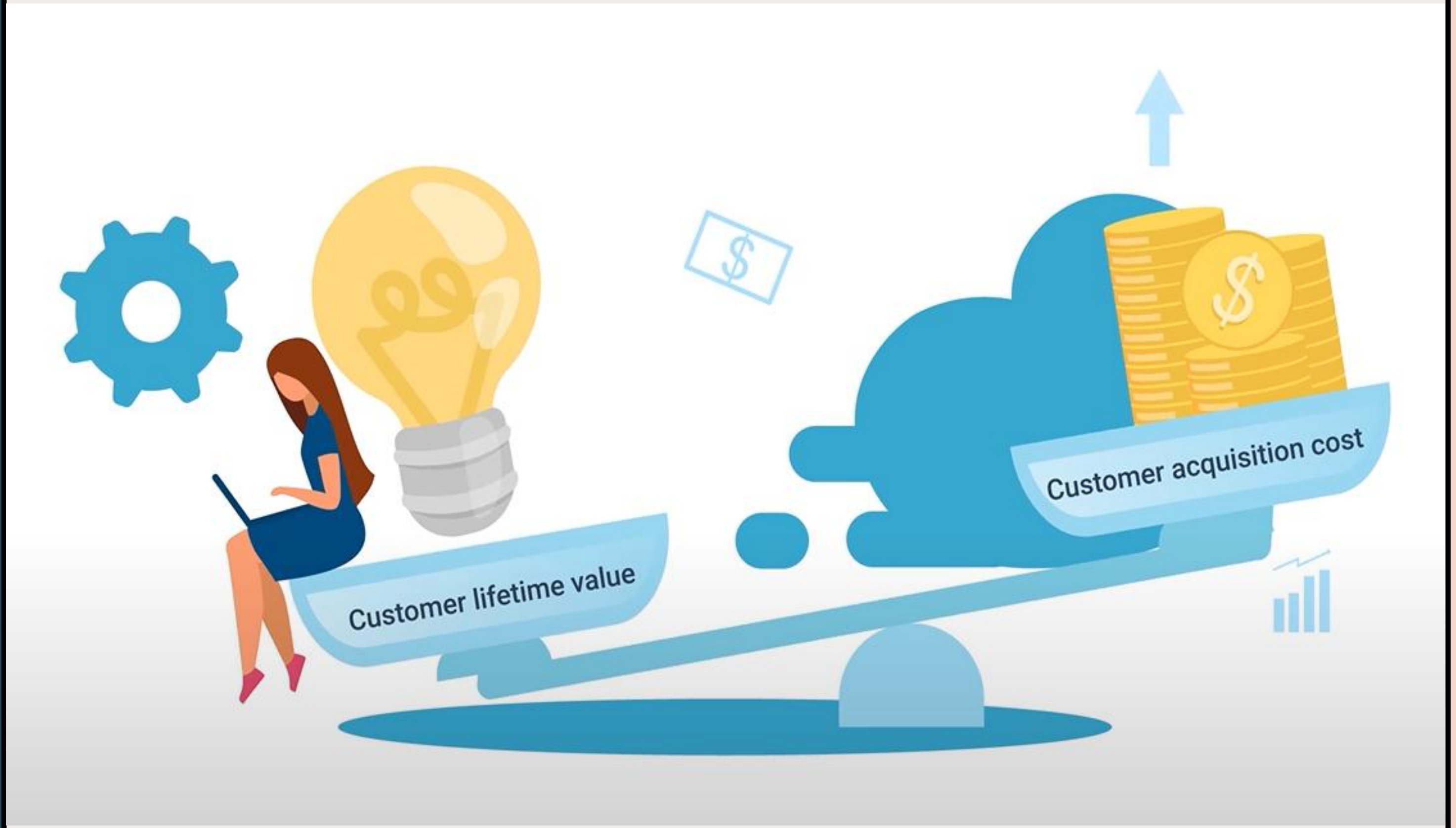
Understanding CLV allows you to make informed decisions based on how long a customer typically buys from you and what they spend over the lifetime of that relationship. This metric can help inform your strategy on acquisition, customer retention, customer support, and even the quality of your products and services.

“

Eighty percent of our business comes from 20% of our customers. It costs 10 times less to sell to an existing customer than to find a new customer

”

Why CLV_



Eighty percent of our business comes from 20% of our customers.
It costs 10 times less to sell to an existing customer than to find a new customer

Businesses prefer retaining customers over acquiring new ones.

Businesses are curious to weigh the value of each customer and determine who is truly worth investing in this is exactly when we use customer lifetime value to identify the total worth of a customer based on their relationship.

Primary goal

Predict a customer's total spending in the next year based on their first few purchases.

Packages Used

Numpy

high-speed mathematical operations

Pandas

for data cleaning, analysis with DataFrames

Scikitlearn

Most imp tool for model training

Openpyxl

Reads, edits, and creates Excel files programmatically

Matplotlib

Turns data into clear visual stories

Plan

Connecting Dots

In the search of
Find a most fitting dataset to perform analysis

Taking useful out of it
Data Cleaning & Analysis of cleaned data

Playing with Data
Model Training using Linear Regression & Random Forest Algorithms

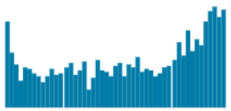
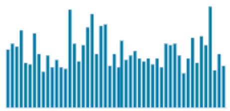
Final Output
Output results, visual analysis of predictions.

Future Scope
Customer Segmentation & Power BI Dashboards

Found a perfect match

data.csv (45.58 MB)

Detail Compact Column 8 of 8 columns

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
25900 unique values	**4070** unique values	**4224** unique values	 -80995 81.0k	 2010-12-01 2011-12-09	-11.1k 39k	 12.3k 18.3k	
536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850	United Kingdom
536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850	United Kingdom
536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850	United Kingdom
536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850	United Kingdom
536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850	United Kingdom
536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12/1/2010 8:26	7.65	17850	United Kingdom
536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	12/1/2010 8:26	4.25	17850	United Kingdom
536366	22633	HAND WARMER UNION JACK	6	12/1/2010 8:28	1.85	17850	United Kingdom

Data Set Overview

Sources:
Online Retail Dataset (UCI ML Repository)
E-commerce Transactions Data (Kaggle)

It is a transnational data set which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail. The company mainly sells unique all-occasion gifts. Many customers of the company are wholesalers.

It contains 8 columns:

InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country

Data Cleaning & Analysis of cleaned data

Imported raw transactional data (data.csv) with **541,909 records**.

```
>>> df = pd.read_csv("data.csv", encoding="latin1")
>>> print(df.head())
  InvoiceNo StockCode  ... CustomerID         Country
0    536365    85123A  ...    17850.0  United Kingdom
1    536365     71053  ...    17850.0  United Kingdom
2    536365    84406B  ...    17850.0  United Kingdom
3    536365    84029G  ...    17850.0  United Kingdom
4    536365    84029E  ...    17850.0  United Kingdom

[5 rows x 8 columns]
>>> df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  object
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

Data Cleaning & Analysis of cleaned data

```
df.dropna(subset=["CustomerID"], inplace=True)
df["InvoiceDate"] = pd.to_datetime(df["InvoiceDate"])
df["TotalSales"] = df["Quantity"] * df["UnitPrice"]
```

Removed missing CustomerID values: removed rows with missing CustomerID
Converted timestamps: Converted **InvoiceDate** into proper datetime format for time-based analysis.
Created TotalSales column: Created a new feature <u>TotalSales = Quantity × UnitPrice</u> for revenue per transaction.

Data Cleaning & Analysis of cleaned data

Final dataset: **406,829 valid transactions** across 8 features + 1 derived feature.

```
>>> df['InvoiceDate']=pd.to_datetime(df['InvoiceDate'])
>>> print(df['InvoiceDate'].dtypes)
datetime64[ns]
>>> df['TotalSales']=df['Quantity']*df['UnitPrice']
>>> df.info()
<class 'pandas.core.frame.DataFrame'>
Index: 406829 entries, 0 to 541908
Data columns (total 9 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    406829 non-null  object
 1   StockCode    406829 non-null  object
 2   Description  406829 non-null  object
 3   Quantity     406829 non-null  int64
 4   InvoiceDate  406829 non-null  datetime64[ns]
 5   UnitPrice    406829 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      406829 non-null  object
 8   TotalSales   406829 non-null  float64
dtypes: datetime64[ns](1), float64(3), int64(1), object(4)
memory usage: 31.0+ MB
>>>
```

How we get there

simple and effective methodology used in calculating customer value over a time frame is RFM_

Recency

R

How recently a customer has made a purchase

Frequency

F

How often a customer makes a purchase

Monetary

M

Dollar value of the purchases

calculating

Recency

Frequency

Monetary

```python
def calculate_rfm(df):
    """Calculate Recency, Frequency, and Monetary metrics."""
    snapshot_date = df["InvoiceDate"].max() + pd.Timedelta(days=1)
    rfm = df.groupby("CustomerID").agg({
        "InvoiceDate": lambda x: (snapshot_date - x.max()).days,
        "InvoiceNo": "nunique",
        "TotalSales": "sum"
    })
```

snapshot date : Sets reference date as day after last transaction in dataset

Groups all rows by CustomerID

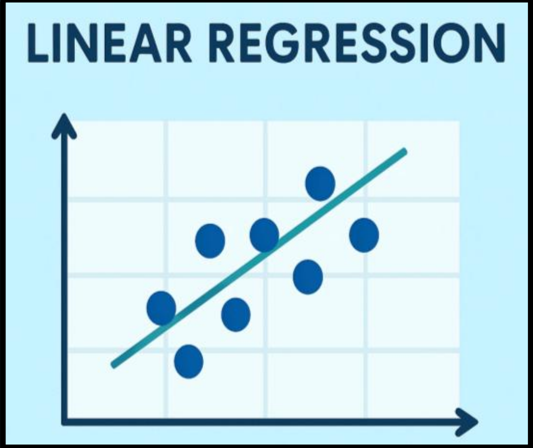
For each customer group, finds their latest purchase date

Calculates days between snapshot and last purchase

Counts number of unique invoice numbers per customer

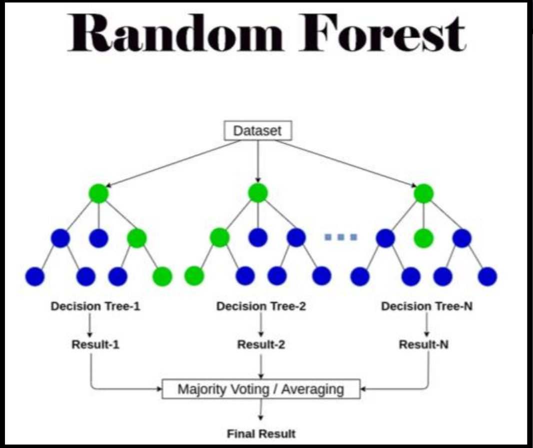
Adds up all Total Sales values per customer.

Model Training



Linear Regression

Model 1



Random Forest

Model 2

```
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
```

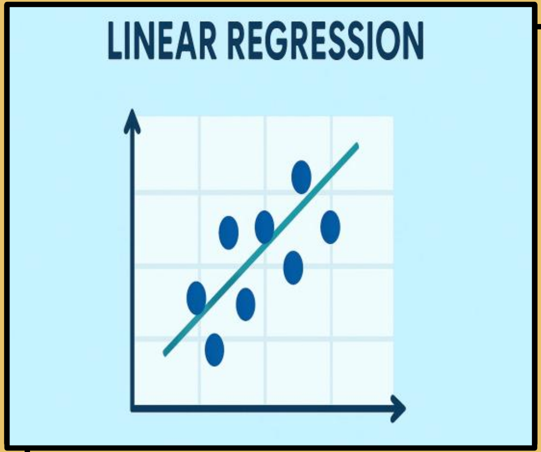
Make essential imports

Then, we divide data in 2 parts_
2/3 for training
1/3 for testing

Models learns the relationship between features & target from 2/3 rd part and predicts the output which is later compared with actual values from 1/3rd part so as to calculate the accuracy.

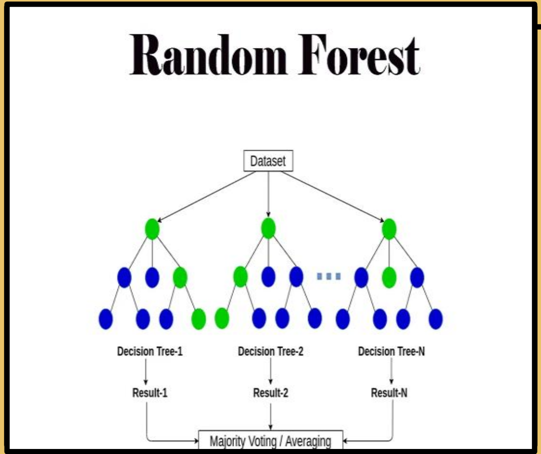
The math is mathing_

Math inside it




```
linreg = LinearRegression().fit(X_train, y_train)
```

FutureMonetary = a×Recency + b×Frequency + c×Monetary + intercept




```
rf = RandomForestRegressor(n_estimators=100, random_state=42).fit(X_train, y_train)
```

Creates 100 decision trees and combines them

$\hat{y} = \beta_0 + \beta_1 R + \beta_2 F + \beta_3 M$; $\beta_1 = ?$, $\beta_2 = ?$; $\beta_3 = ?$

To solve this, we need to solve_

$\beta = (X^T X)^{-1} X^T y$

where β = coeff. vector

y = target vector (actual future Monetory from test data set)

$X = [\beta_0 \quad R \quad F \quad M]$ for each customer

β_0 → constant for all (let = 1)

The linear model : $\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$

∴ in matrix form: $\hat{y} = X\beta$

$$[\quad]_{1\times1} = [\beta_0 \, R F M]_{1\times4} * \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}_{4\times1}$$

To minimized sum of squared Errors_

cost function: $J(\beta) = \Sigma (y_i - \hat{y}_i)^2 = (y - X\beta)^T (y - X\beta)$

$$= (y^T - \beta^T X^T)(y - X\beta)$$

$$= y^T y - 2\beta^T X^T y + \beta^T X^T X \beta$$

$$\frac{\partial J\beta}{\partial \beta} = \frac{\partial}{\partial \beta}\left(y^T y - 2\beta^T X^T y + \beta^T X^T X \beta\right) = 0$$

$$\Rightarrow \quad -2X^T y + 2X^T X\beta = 0$$

solve for β= $\quad X^T y = X^T X \beta$

$$\therefore \beta = (X^T X)^{-1} X^T y$$

Example : suppose after training we get

$\beta_0 = 50.0$

$\beta_1 = -2.0$

$\beta_2 = 25.0$

$\beta_3 = 0.8$

predicted formula_

$\therefore \hat{y} = 50.0 + (-2.0)R + (25.0)F + (0.8)M$

lr

n_estimators = 100 ⇒ creates 100 decision trees

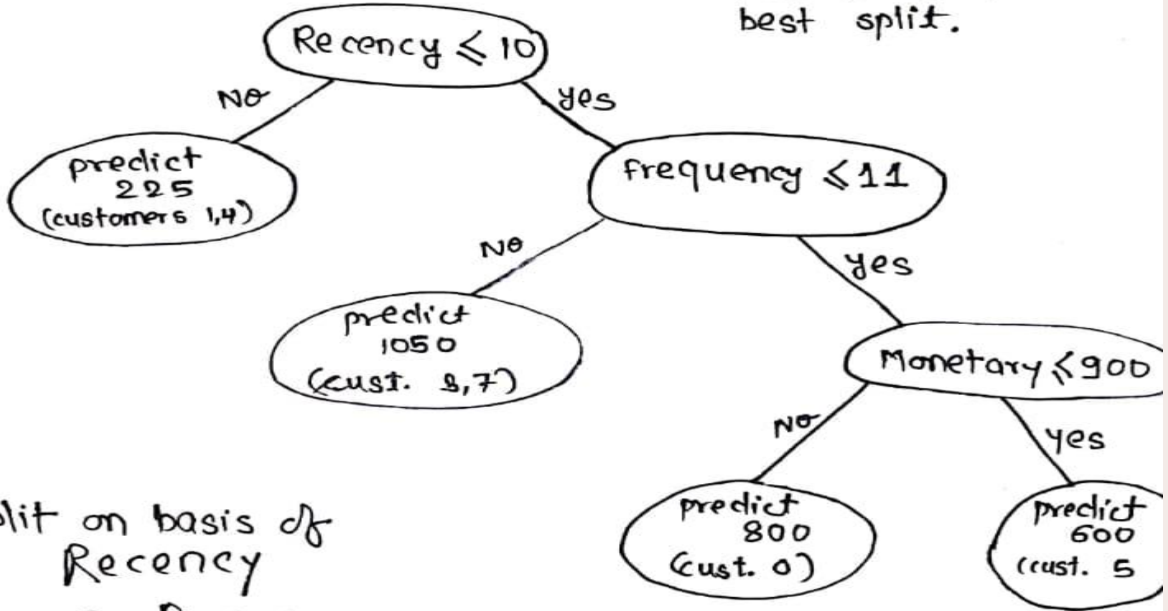
⇒ Random Forest creates 100 different training subsets

Tree 1 : [0, 1, 3, 4, 5, 7]
Tree 2 : [2, 3, 4, 5, 6, 7] → customers
Tree 3 : [0, 1, 2, 5, 6, 7]
⋮

onsider Tree 1 = ALGORITHM tries all possible splits across all features.

complete Tree 1 structure.

Root Node aims for best split.



first split on basis of Recency
then on Frequency
then on Monetary.

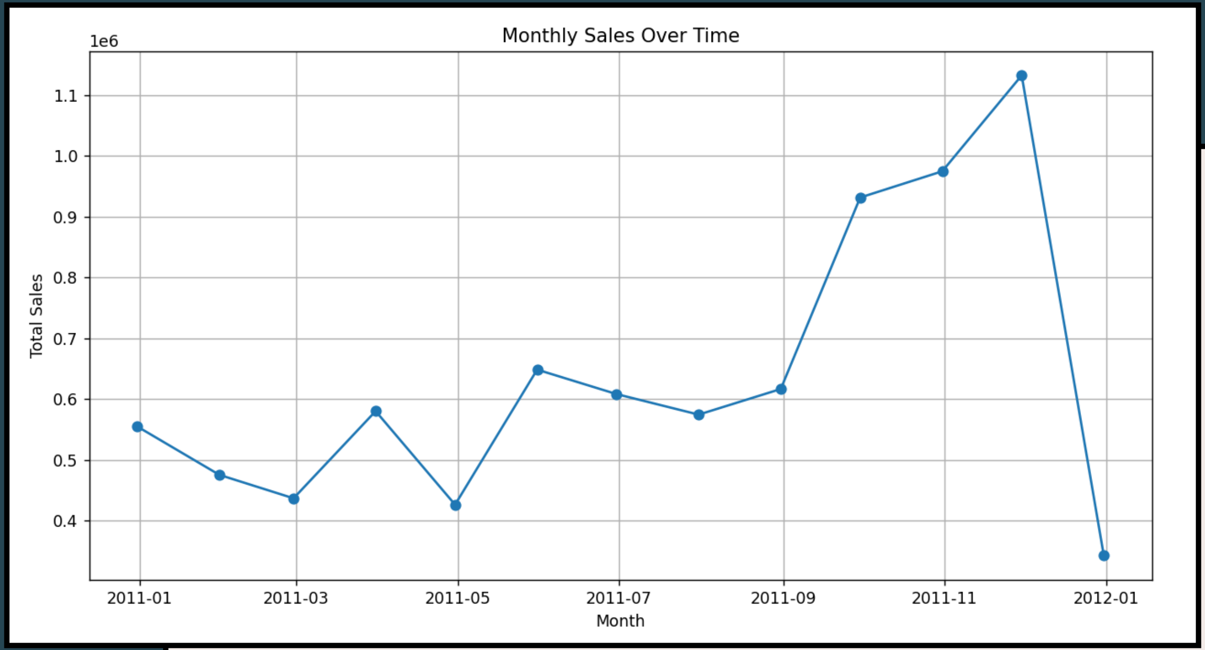
ree 2, 3 ... 100 splits vary. splits will also change the order

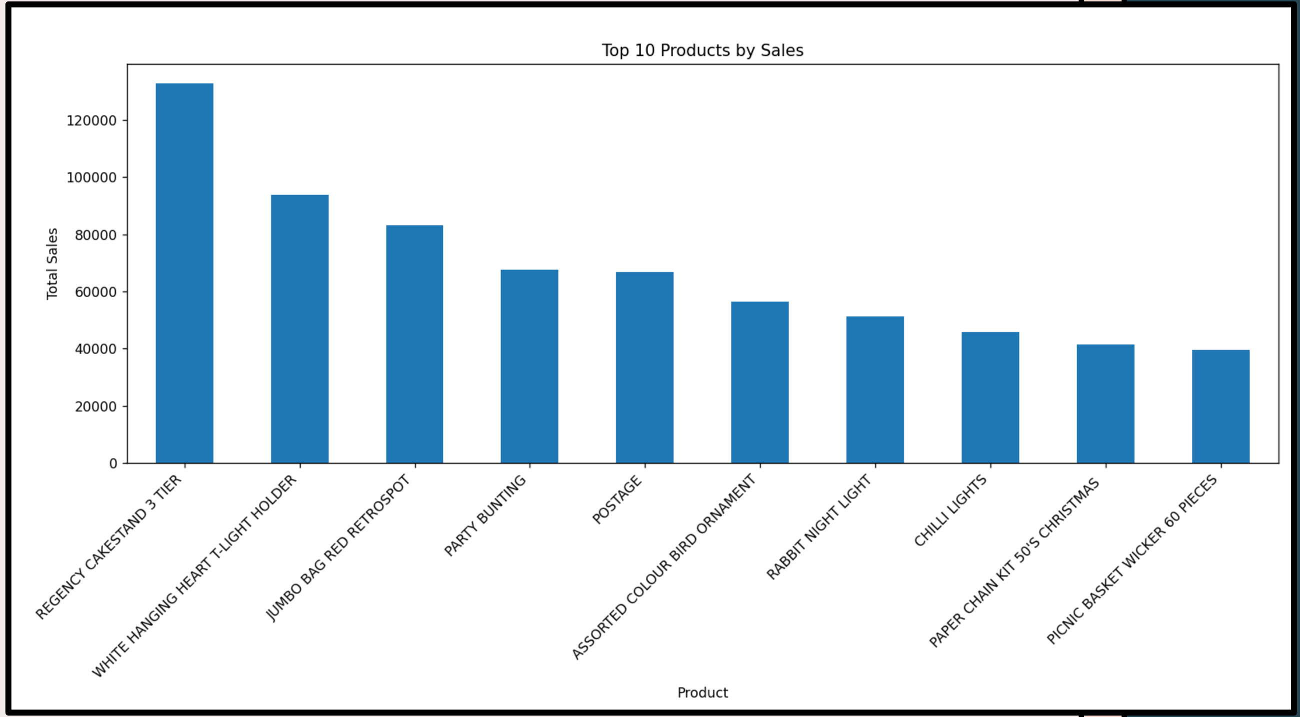
Hence, Final prediction $= \left(\frac{1}{100}\right) * \sum \text{Tree Prediction}_i$

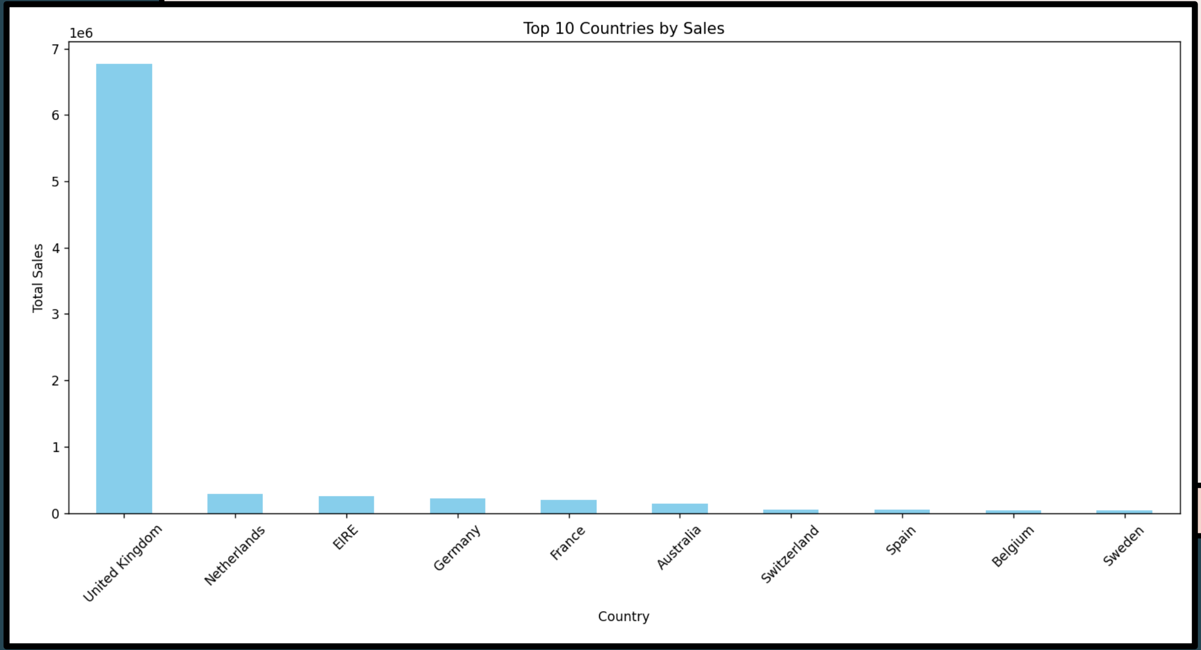
rf

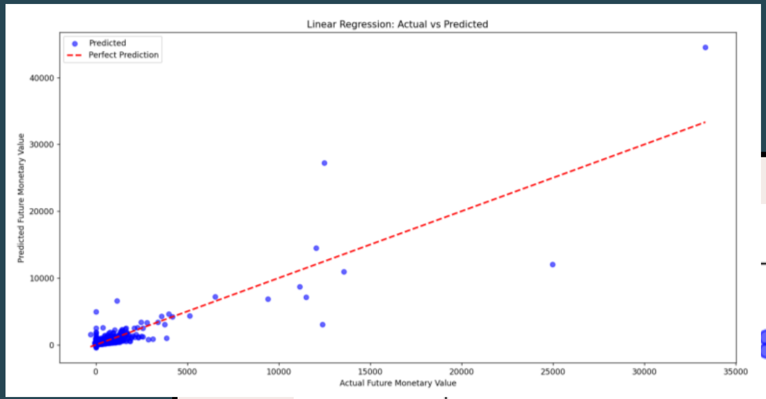
Final outputs_

plot_monthly_sales,
plot_top_products,
plot_top_countries,
plot_predictions(y_test, y_pred_lr, "Linear Regression", "blue")
plot_predictions(y_test, y_pred_rf, "Random Forest", "green")

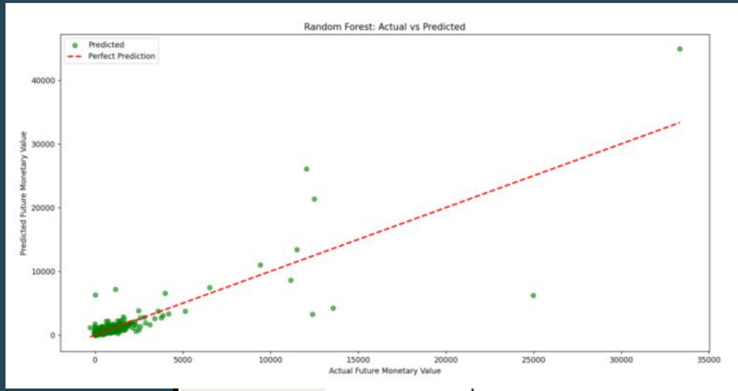

Monthly Sales Over Time
1e6
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
Total Sales
2011-01
2011-03
2011-05
2011-07
2011-09
2011-11
2012-01
Month

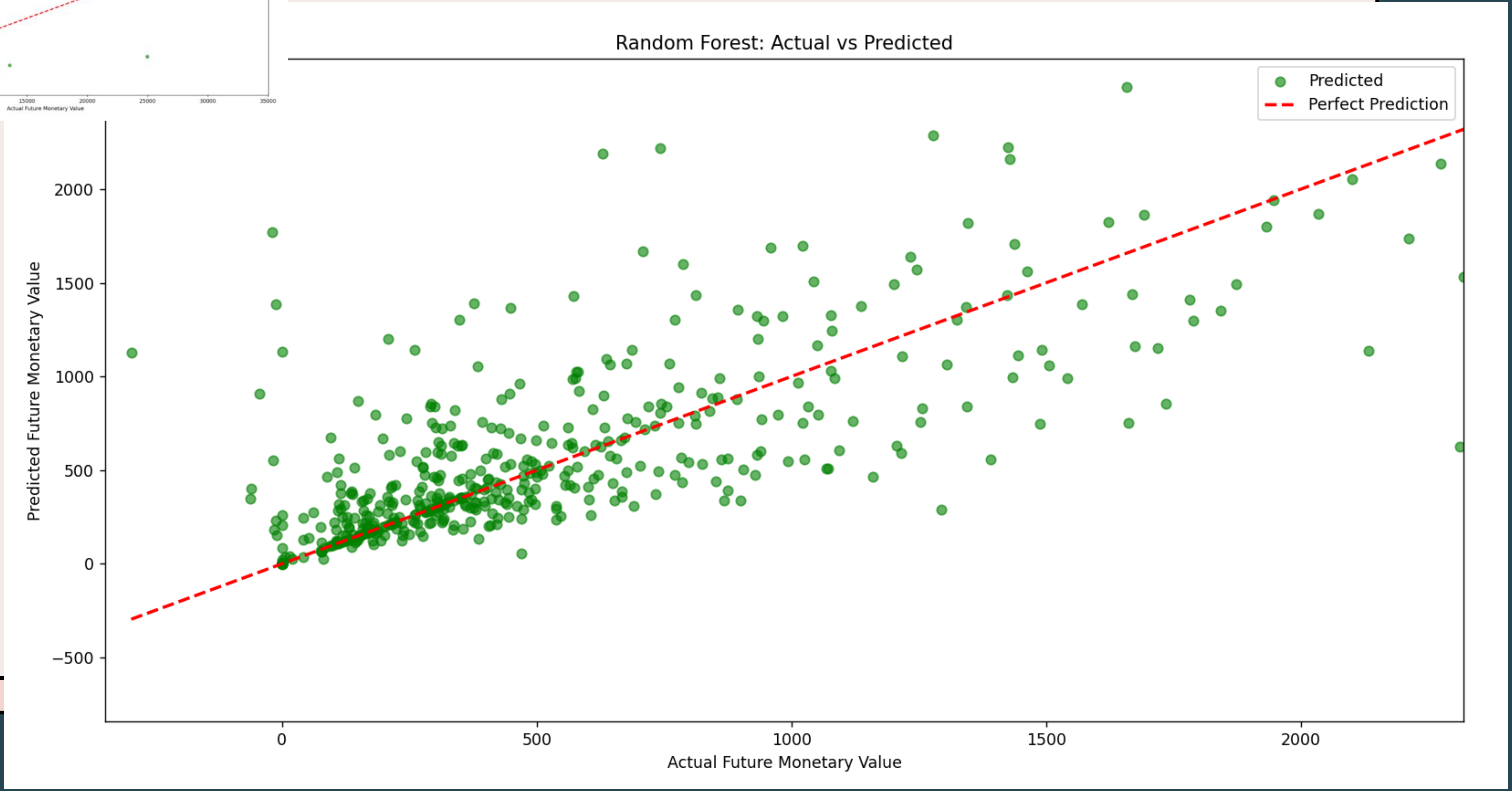

Top 10 Products by Sales
120000
100000
80000
60000
40000
20000
0
Total Sales
REGENCY CAKESTAND 3 TIER
WHITE HANGING HEART T-LIGHT HOLDER
JUMBO BAG RED RETROSPOT
PARTY BUNTING
POSTAGE
ASSORTED COLOUR BIRD ORNAMENT
RABBIT NIGHT LIGHT
CHILLI LIGHTS
PAPER CHAIN KIT 50'S CHRISTMAS
PICNIC BASKET WICKER 60 PIECES
Product


Top 10 Countries by Sales
1e6
7
6
5
4
3
2
1
0
Total Sales
United Kingdom
Netherlands
EIRE
Germany
France
Australia
Switzerland
Spain
Belgium
Sweden
Country


Linear Regression: Actual vs Predicted
Predicted
Perfect Prediction
Predicted Future Monetary Value
Actual Future Monetary Value


Linear Regression: Actual vs Predicted
Predicted Future Monetary Value
Actual Future Monetary Value
1500
1000
500
0
−500
−250
0
250
500
750
1000
1250
1500


Random Forest: Actual vs Predicted
Predicted
Perfect Prediction
Predicted Future Monetary Value
Actual Future Monetary Value
0
10000
20000
30000
40000
5000
15000
25000
35000


Random Forest: Actual vs Predicted
Predicted
Perfect Prediction
Predicted Future Monetary Value
Actual Future Monetary Value
2000
1500
1000
500
0
−500
0
500
1000
1500
2000

Future scope

Create an **interactive dashboard (Tableau/Power BI)** for marketing managers to visualize CLV predictions in real-time.

Use unsupervised learning (K-Means) on the RFM features to identify **5-7 distinct customer segments** for highly targeted marketing.

Thank you.