**A Data-Driven Approach for Customer Lifetime Value (CLV) in E-Commerce**

30% code

import numpy as np

import pandas as pd

import joblib

# Visualization

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn.preprocessing import LabelEncoder

#Scaling

from sklearn.preprocessing import StandardScaler

#Train Test Split

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

# Models

from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.svm import SVC

#Evaluation

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

import os

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder

import seaborn as sns

import warnings

warnings.filterwarnings("ignore")

from sklearn.utils import resample

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

import os

import joblib

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score,confusion\_matrix,classification\_report,precision\_score,recall\_score,f1\_score

#pip install openpyxl

df = pd.read\_csv(r'Datasets/Year 2009-2010.csv',encoding="latin1")

df.head()

df['Country'].unique()

df.size

df.shape

df.describe()

df.dtypes

df.columns

df.isnull().sum()

df.dropna(inplace = True)

df.duplicated().sum()

df.drop\_duplicates(inplace = True)

unique\_counts = []

for col in df: # You can add more column names here if needed

unique\_count = df[col].nunique()

unique\_counts.append((col, unique\_count))

print(col,"=" ,unique\_count)

df.corr()

plt.figure(figsize = (18,6))

sns.countplot(data = df)

sns.heatmap(df.corr(), annot = True)

plt.title('Correlation Heatmap')

plt.show()

X = df.drop(['Quantity'],axis = 1)

X

X.info()

y = df['Quantity']

y

X\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.2, random\_state = 42)

**EXISTING SYSTEM**

Traditional systems for CLV estimation, such as RFM analysis and rule-based segmentation, are not dynamic and fail to account for new customer trends. Manual surveys and CRM data analysis are slow and may yield incomplete or biased insights. Historical sales-based forecasting is often too simplistic, unable to predict future behaviors with accuracy. These methods are not scalable and do not adapt to shifting customer patterns or market conditions, leading to poor personalization and missed revenue opportunities.

**DISADVANTAGES**

* **Lack of Real-Time Adaptability** – Traditional methods like RFM (Recency, Frequency, Monetary) and rule-based segmentation cannot adjust dynamically to changing customer behaviors, trends, or market conditions.
* **Oversimplification of Customer Behavior** – Historical sales-based forecasting assumes that past behavior predicts future behavior, ignoring factors like changing preferences, competition, and economic conditions.
* **Inability to Personalize Effectively** – Since traditional methods rely on broad segmentation, they fail to offer personalized experiences, leading to generic marketing strategies that may not resonate with individual customers.
* **Slow and Resource-Intensive** – Manual surveys and CRM data analysis take time, and by the time insights are generated, they may already be outdated, making them ineffective for agile decision-making.
* **Bias and Incompleteness** – Surveys often suffer from response bias, and CRM data may lack crucial external factors (e.g., competitor influence, macroeconomic trends), leading to inaccurate CLV predictions.
* **Limited Scalability** – These methods struggle to handle large and complex datasets, making them impractical for businesses with a growing customer base or diverse product offerings.
* **Missed Revenue Opportunities** – Due to their static nature and lack of predictive power, traditional CLV models fail to identify high-value customers early, resulting in inefficient marketing spend and lost revenue potential.

**PROPOSED METHODOLOGY**

**Decision Tree Regression**

Decision Tree Regression is a machine learning algorithm that predicts a continuous target variable by splitting the dataset into smaller and smaller subsets based on decision rules. It works similarly to a flowchart, where each internal node represents a feature (predictor), each branch represents a decision rule, and each leaf node represents a predicted value.

**How Decision Tree Regression Works**

1. **Splitting the Data**
   * The algorithm selects the best feature and threshold to split the data into two or more groups.
   * The split is chosen to minimize the variance (or another impurity measure) within the resulting subsets.
2. **Recursive Partitioning**
   * The process is repeated recursively for each subset until a stopping condition is met (e.g., max depth, min samples per leaf).
3. **Assigning Predictions**
   * Once the tree reaches a leaf node, it assigns the average value of the target variable within that node as the prediction.

**Advantages of Decision Tree Regression**

**Easy to Understand & Interpret** – Simple, rule-based structure.  
**Non-Linear Relationships** – Can model complex relationships between features and the target variable.  
**No Need for Feature Scaling** – Unlike linear regression, it does not require normalization or standardization.

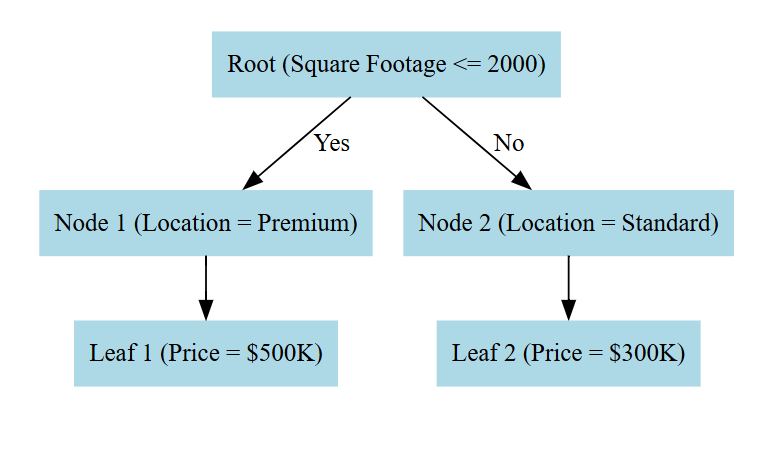


Fig 1: Block Diagram of Decision Tree Regression

**Lasso Regression**

Lasso Regression is a type of linear regression that includes L1 regularization to prevent overfitting and perform feature selection. It helps simplify models by shrinking some coefficients to exactly zero, effectively removing less important features.

**How Lasso Regression Works**

1. **Starts Like Linear Regression**
   * Lasso regression builds a linear model:



Here, y is the target variable, x1, x2, ..., xn are the features, and β\betaβ are the coefficients.

1. **Adds L1 Regularization (Penalty Term)**
   * Unlike simple linear regression, Lasso minimizes:



* + The second term is the L1 penalty, where λ (lambda) is a tuning parameter that controls the amount of regularization.

1. **Feature Selection Effect**
   * When λ is large, some coefficients become exactly zero, eliminating irrelevant features.
   * This makes Lasso useful for selecting only the most important variables.

**Advantages of Lasso Regression**

* **Feature Selection** – Automatically removes unimportant variables.
* **Prevents Overfitting** – Reduces model complexity by shrinking coefficients.
* **Works Well for High-Dimensional Data** – Useful when there are many features.

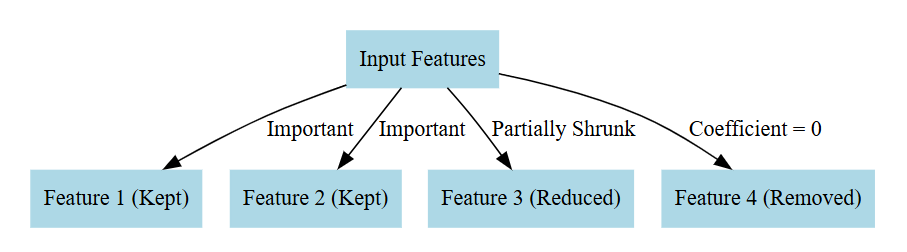


Fig 2: Block Diagram of Lasso Regression

**BLOCK DIAGRAM:**

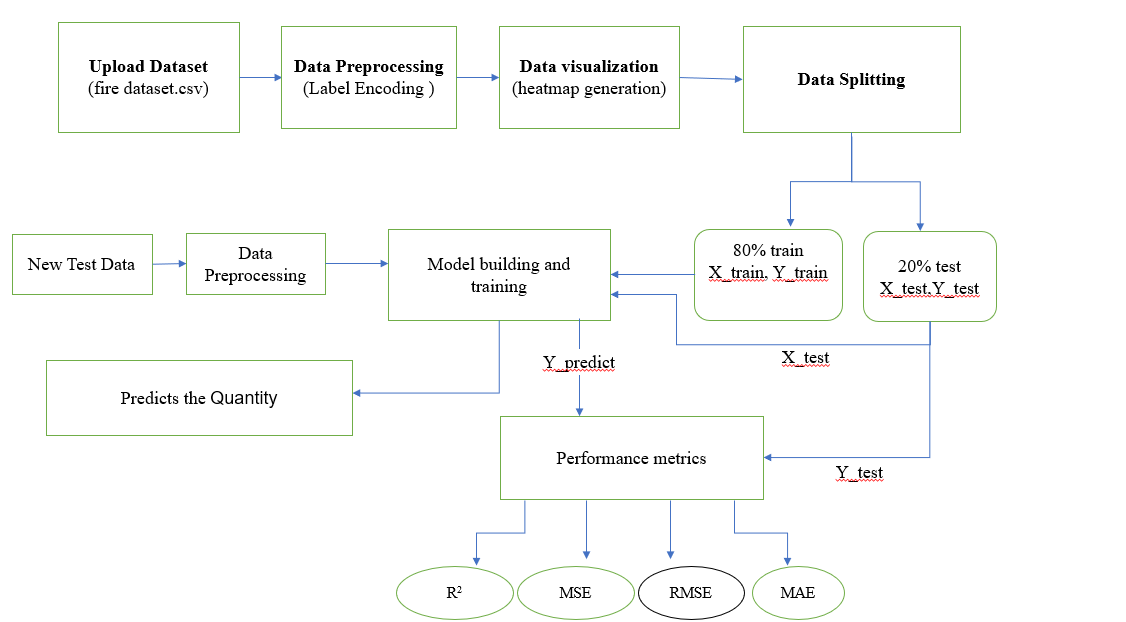
****

Fig 3: Proposed Block Diagram

**EXPECTED OUTPUT:**

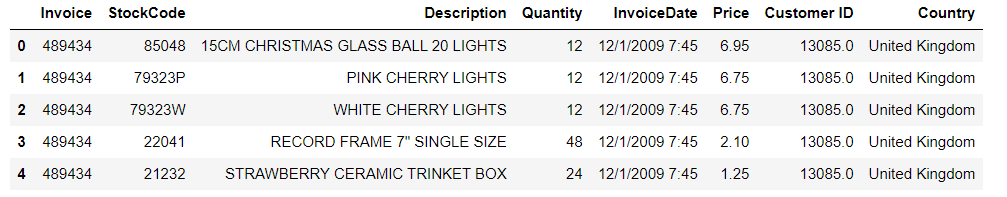
****

Fig 4: Uploading Dataset

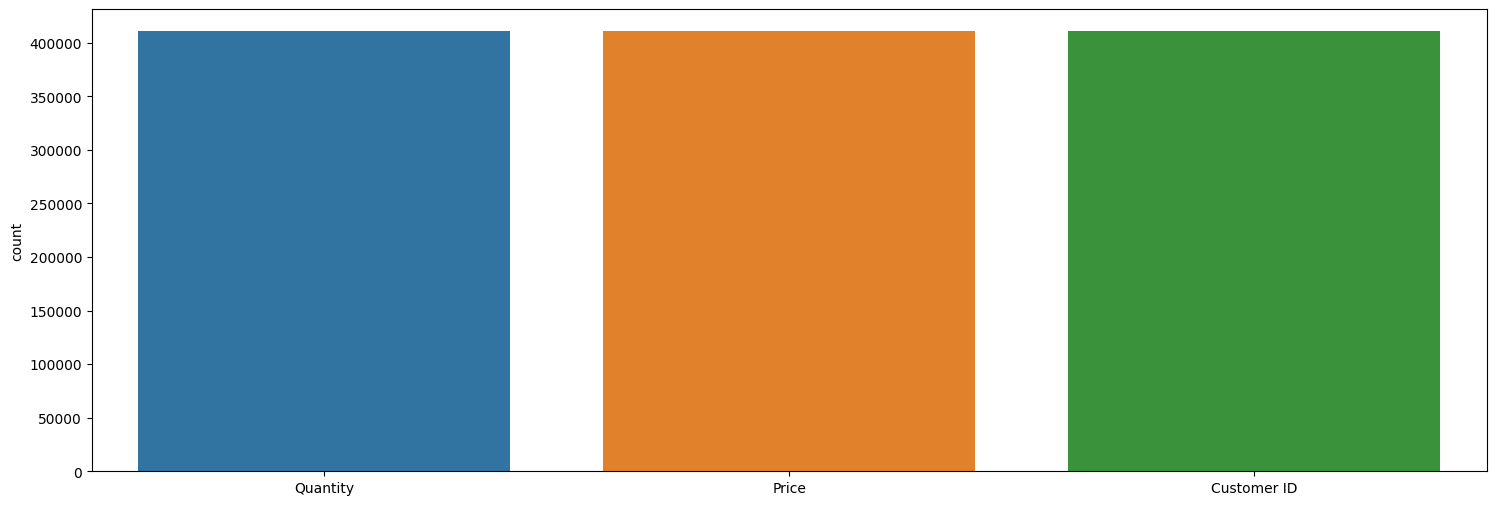


Fig 5: Count plot

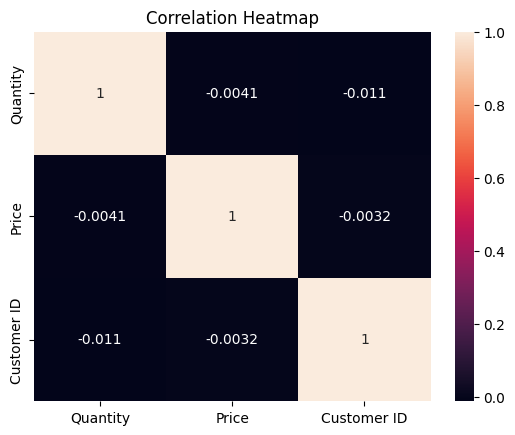
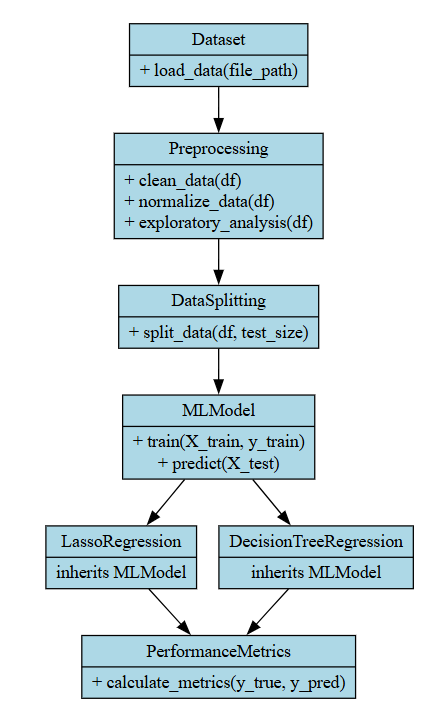


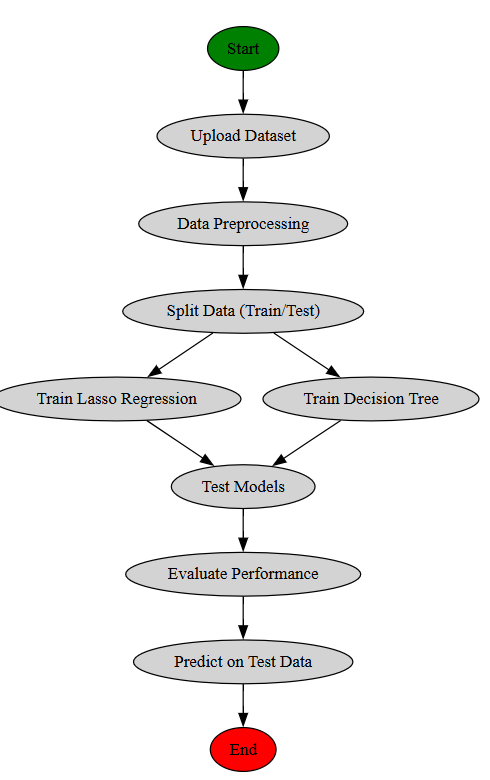
Fig 6: Heat map

**UML DIAGRAMS**

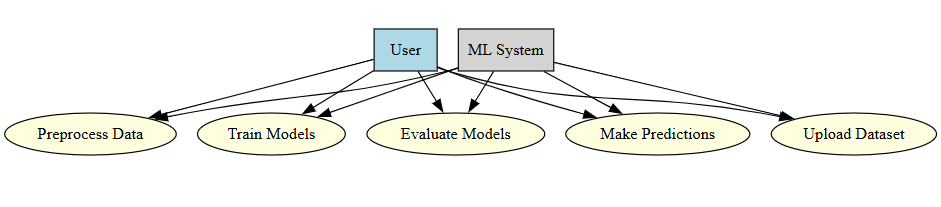
**1. Class Diagram**



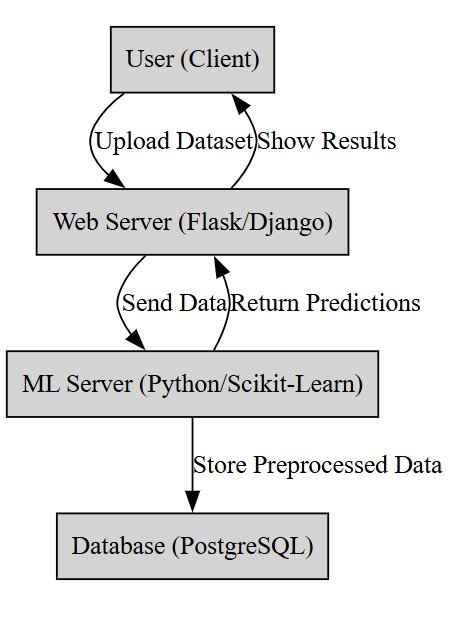
**2. Activity Diagram**

****

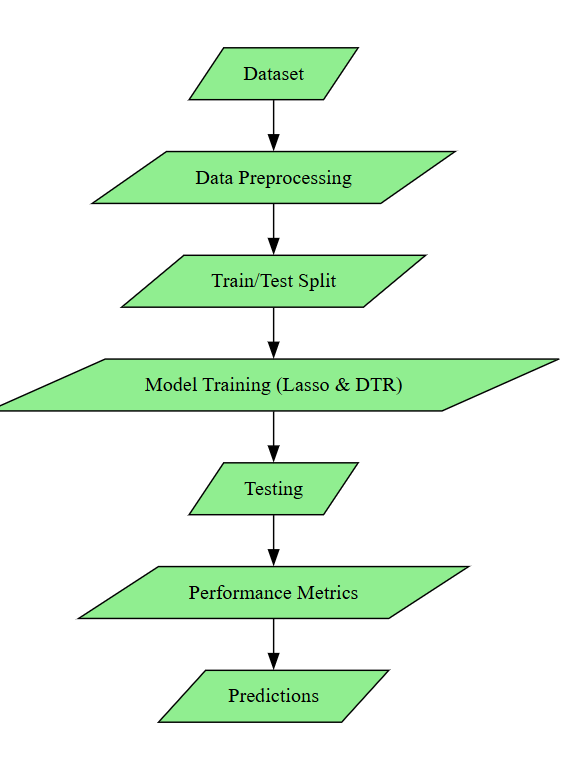
**3. Use Case Diagram**



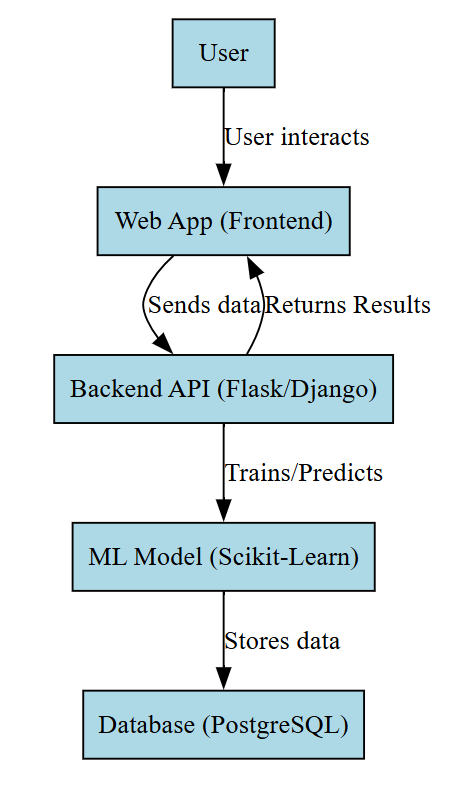
**4. Deployment Diagram**

****

**5. Data Flow Diagram**

****

**6. Architectural Block Diagram**

****