IMPROVING DATA ACCURACY IN CRM USING AI

PHASE 4

Submitted by:

Aditya Mahesh Patil	2VD21CS001	Candidate Id: CAN_33858141
Amarnath Mahesh Patil	2VD21CS006	Candidate Id: CAN_33858144
Hrishikesh M Agnihotri	2VD21CS018	Candidate Id: CAN_33887162
Nivedita G Nayak	2VD21CS033	Candidate Id: CAN_33699847

1. Overview of the Project Results

This project focuses on enhancing the accuracy and reliability of Customer Relationship Management (CRM) data using AI-driven automation. The system detects and removes duplicate records, imputes missing values using predictive algorithms, and ensures consistency, ultimately improving data quality for better decision-making and business insights. By leveraging AI, this project minimizes human intervention and enhances the efficiency of CRM systems used by organizations

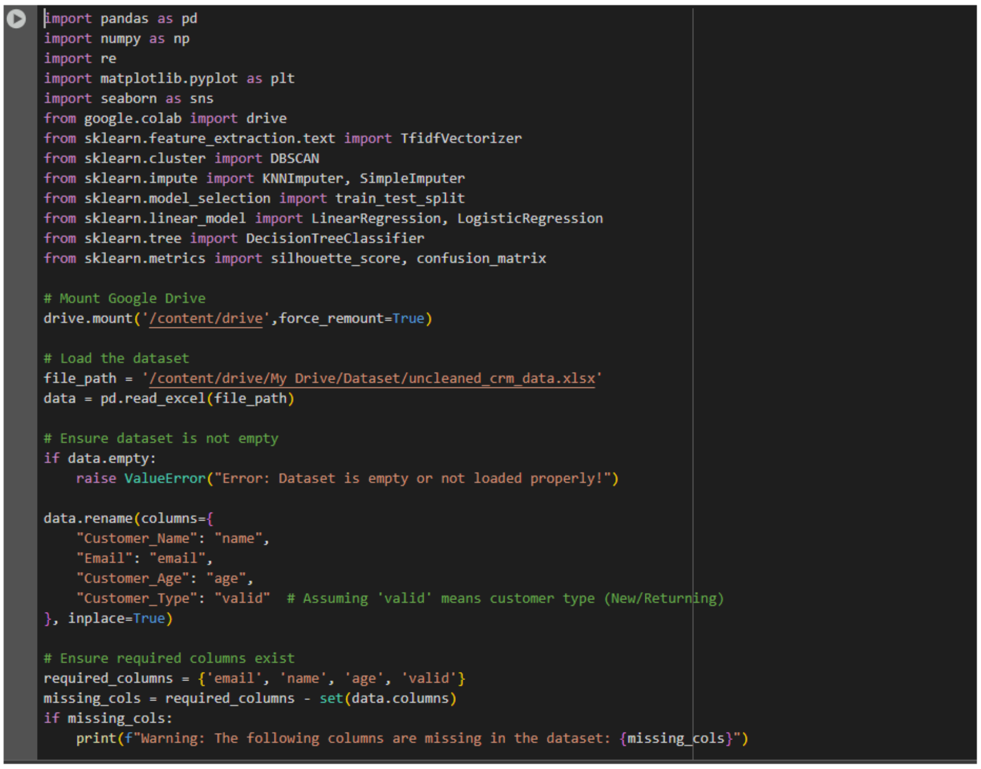
2. Key Features

- Data Cleaning & Standardization – Detects and corrects inconsistencies in CRM data, ensuring uniform formats.
- Duplicate Detection – Identifies and removes redundant customer records to maintain data integrity.
- Missing Value Imputation – Uses AI models to predict and fill in missing data points, reducing data gaps.
- Data Quality Assessment – Generates reports and visualizations to evaluate improvements in data accuracy.

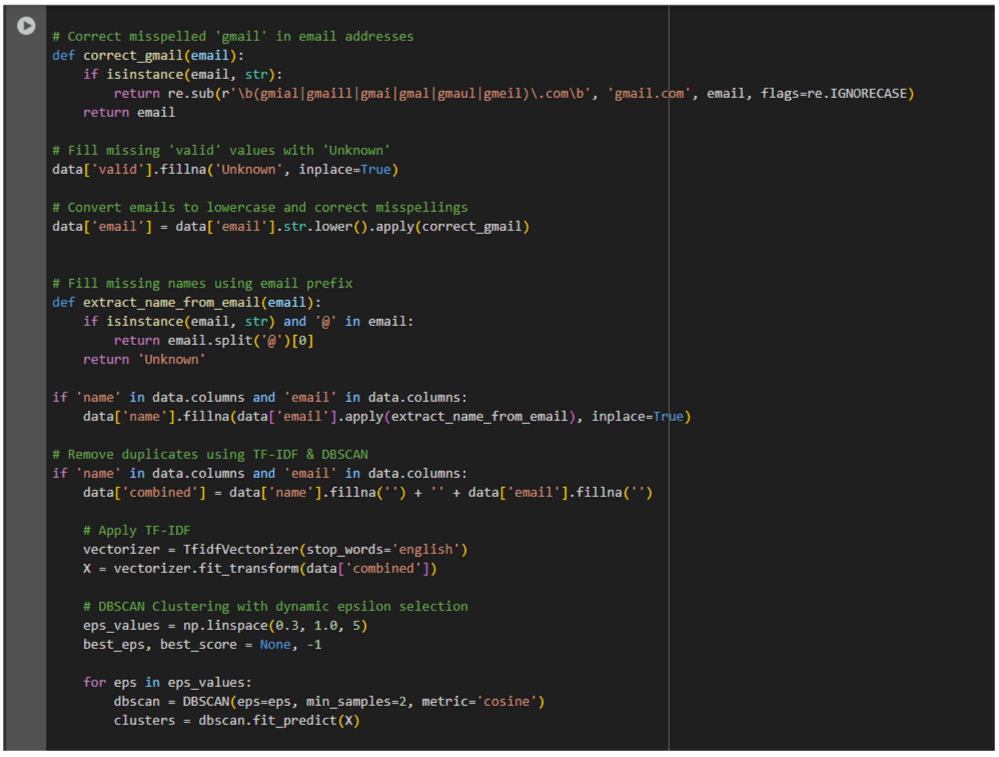
3. Tech Stack

- Programming Language: Python – Chosen for its rich ecosystem in data science and machine learning.
- Libraries: Pandas (data processing), NumPy (numerical operations), Scikit-learn (machine learning models), Matplotlib & Seaborn (data visualization).
- Machine Learning Models:
    - DBSCAN: Detects duplicate records by clustering similar entries.
    - KNN Imputer: Fills in missing values based on the nearest neighbors' data.
    - Decision Trees: Identifies and corrects anomalies in CRM datasets.
- Development Environment: Google Colab – Used for coding, training models, and visualizing results.

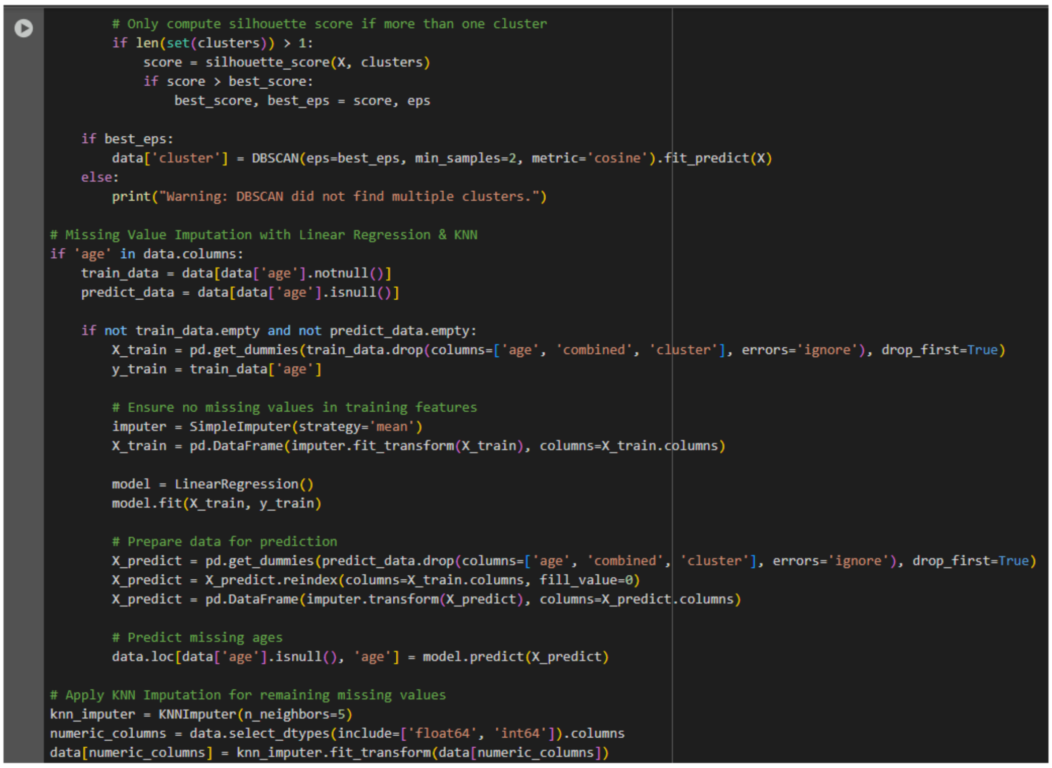
Data Cleaning and preprocessing



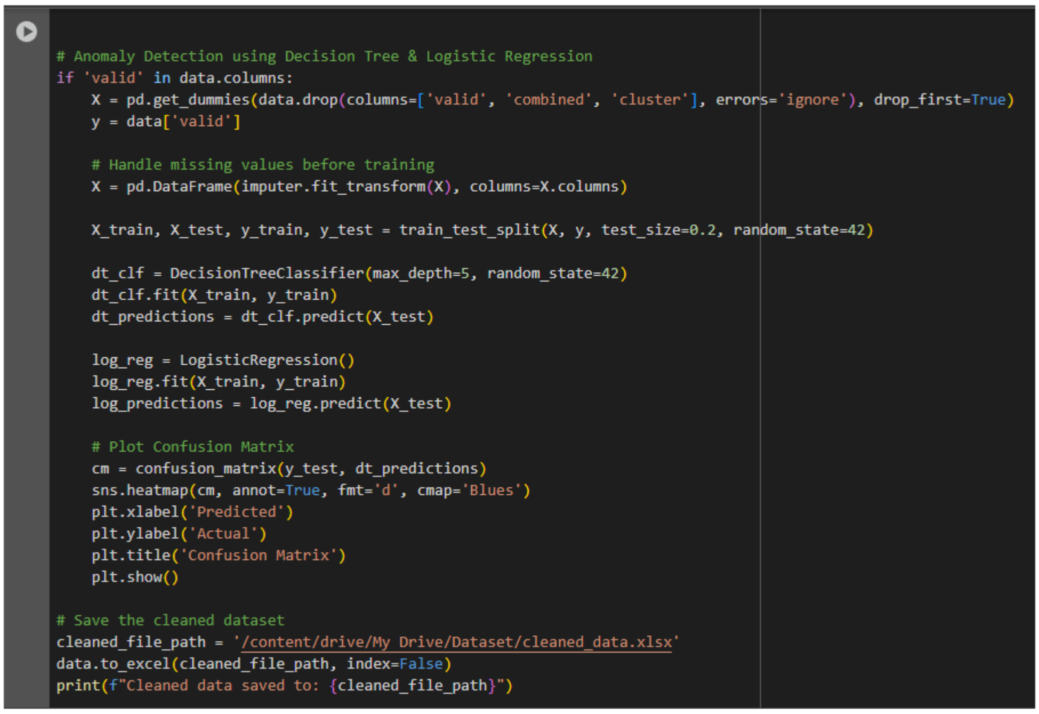
```python
import pandas as pd
import numpy as np
import re
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import drive
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import DBSCAN
from sklearn.impute import KNNImputer, SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import silhouette_score, confusion_matrix

# Mount Google Drive
drive.mount('/content/drive',force_remount=True)

# Load the dataset
file_path = '/content/drive/My Drive/Dataset/uncleaned_crm_data.xlsx'
data = pd.read_excel(file_path)

# Ensure dataset is not empty
if data.empty:
    raise ValueError("Error: Dataset is empty or not loaded properly!")

data.rename(columns={
    "Customer_Name": "name",
    "Email": "email",
    "Customer_Age": "age",
    "Customer_Type": "valid"  # Assuming 'valid' means customer type (New/Returning)
}, inplace=True)

# Ensure required columns exist
required_columns = {'email', 'name', 'age', 'valid'}
missing_cols = required_columns - set(data.columns)
if missing_cols:
    print(f"Warning: The following columns are missing in the dataset: {missing_cols}")
```



```python
# Correct misspelled 'gmail' in email addresses
def correct_gmail(email):
    if isinstance(email, str):
        return re.sub(r'\b(gmial|gmaill|gmai|gmal|gmaul|gmeil)\.com\b', 'gmail.com', email, flags=re.IGNORECASE)
    return email

# Fill missing 'valid' values with 'Unknown'
data['valid'].fillna('Unknown', inplace=True)

# Convert emails to lowercase and correct misspellings
data['email'] = data['email'].str.lower().apply(correct_gmail)


# Fill missing names using email prefix
def extract_name_from_email(email):
    if isinstance(email, str) and '@' in email:
        return email.split('@')[0]
    return 'Unknown'

if 'name' in data.columns and 'email' in data.columns:
    data['name'].fillna(data['email'].apply(extract_name_from_email), inplace=True)

# Remove duplicates using TF-IDF & DBSCAN
if 'name' in data.columns and 'email' in data.columns:
    data['combined'] = data['name'].fillna('') + '' + data['email'].fillna('')

    # Apply TF-IDF
    vectorizer = TfidfVectorizer(stop_words='english')
    X = vectorizer.fit_transform(data['combined'])

    # DBSCAN Clustering with dynamic epsilon selection
    eps_values = np.linspace(0.3, 1.0, 5)
    best_eps, best_score = None, -1

    for eps in eps_values:
        dbscan = DBSCAN(eps=eps, min_samples=2, metric='cosine')
        clusters = dbscan.fit_predict(X)
```

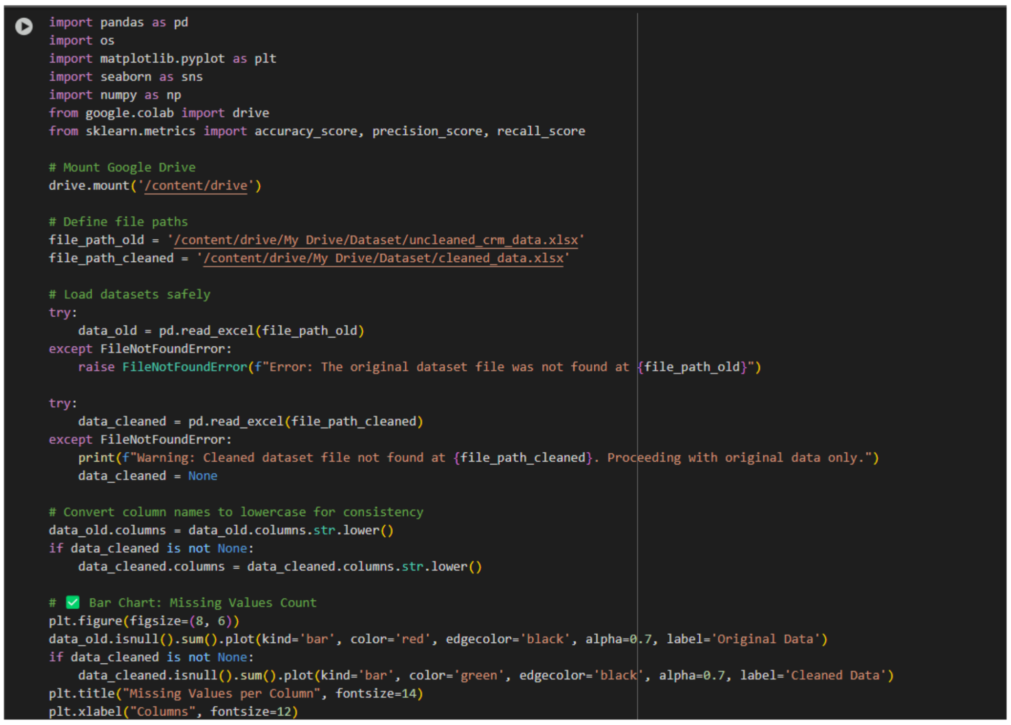


```python
        # Only compute silhouette score if more than one cluster
        if len(set(clusters)) > 1:
            score = silhouette_score(X, clusters)
            if score > best_score:
                best_score, best_eps = score, eps

    if best_eps:
        data['cluster'] = DBSCAN(eps=best_eps, min_samples=2, metric='cosine').fit_predict(X)
    else:
        print("Warning: DBSCAN did not find multiple clusters.")

# Missing Value Imputation with Linear Regression & KNN
if 'age' in data.columns:
    train_data = data[data['age'].notnull()]
    predict_data = data[data['age'].isnull()]

    if not train_data.empty and not predict_data.empty:
        X_train = pd.get_dummies(train_data.drop(columns=['age', 'combined', 'cluster'], errors='ignore'), drop_first=True)
        y_train = train_data['age']

        # Ensure no missing values in training features
        imputer = SimpleImputer(strategy='mean')
        X_train = pd.DataFrame(imputer.fit_transform(X_train), columns=X_train.columns)

        model = LinearRegression()
        model.fit(X_train, y_train)

        # Prepare data for prediction
        X_predict = pd.get_dummies(predict_data.drop(columns=['age', 'combined', 'cluster'], errors='ignore'), drop_first=True)
        X_predict = X_predict.reindex(columns=X_train.columns, fill_value=0)
        X_predict = pd.DataFrame(imputer.transform(X_predict), columns=X_predict.columns)

        # Predict missing ages
        data.loc[data['age'].isnull(), 'age'] = model.predict(X_predict)

# Apply KNN Imputation for remaining missing values
knn_imputer = KNNImputer(n_neighbors=5)
numeric_columns = data.select_dtypes(include=['float64', 'int64']).columns
data[numeric_columns] = knn_imputer.fit_transform(data[numeric_columns])
```

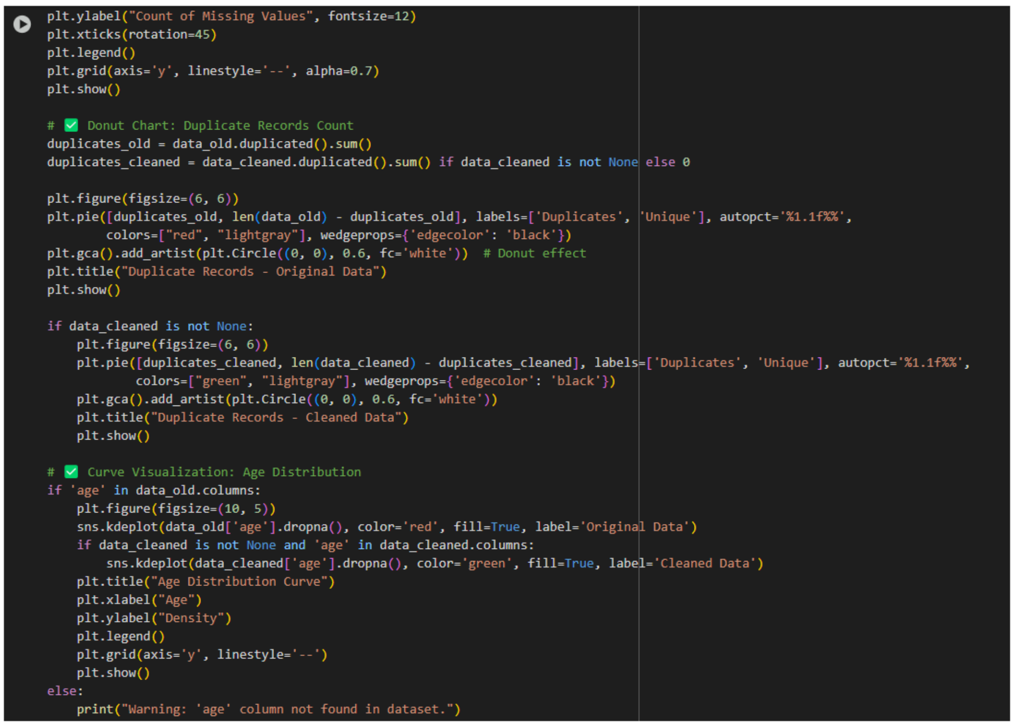


```python
# Anomaly Detection using Decision Tree & Logistic Regression
if 'valid' in data.columns:
    X = pd.get_dummies(data.drop(columns=['valid', 'combined', 'cluster'], errors='ignore'), drop_first=True)
    y = data['valid']

    # Handle missing values before training
    X = pd.DataFrame(imputer.fit_transform(X), columns=X.columns)

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    dt_clf = DecisionTreeClassifier(max_depth=5, random_state=42)
    dt_clf.fit(X_train, y_train)
    dt_predictions = dt_clf.predict(X_test)

    log_reg = LogisticRegression()
    log_reg.fit(X_train, y_train)
    log_predictions = log_reg.predict(X_test)

    # Plot Confusion Matrix
    cm = confusion_matrix(y_test, dt_predictions)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title('Confusion Matrix')
    plt.show()

# Save the cleaned dataset
cleaned_file_path = '/content/drive/My Drive/Dataset/cleaned_data.xlsx'
data.to_excel(cleaned_file_path, index=False)
print(f"Cleaned data saved to: {cleaned_file_path}")
```

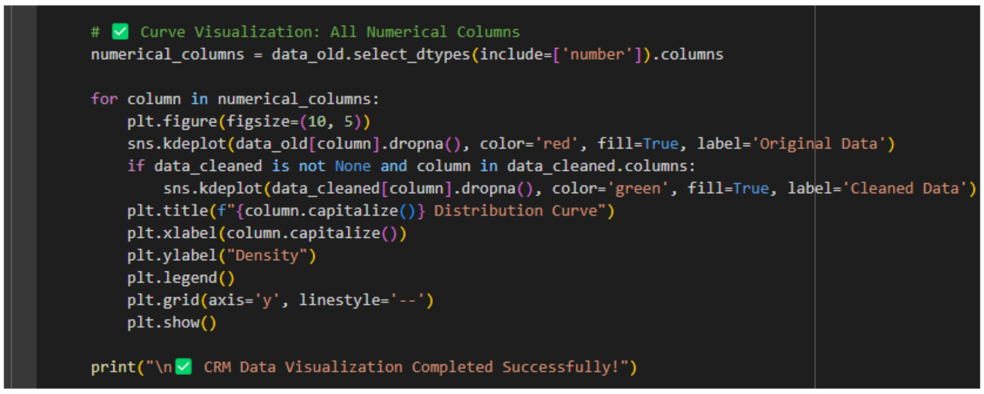
Data Visualization



```python
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from google.colab import drive
from sklearn.metrics import accuracy_score, precision_score, recall_score

# Mount Google Drive
drive.mount('/content/drive')

# Define file paths
file_path_old = '/content/drive/My Drive/Dataset/uncleaned_crm_data.xlsx'
file_path_cleaned = '/content/drive/My Drive/Dataset/cleaned_data.xlsx'

# Load datasets safely
try:
    data_old = pd.read_excel(file_path_old)
except FileNotFoundError:
    raise FileNotFoundError(f"Error: The original dataset file was not found at {file_path_old}")

try:
    data_cleaned = pd.read_excel(file_path_cleaned)
except FileNotFoundError:
    print(f"Warning: Cleaned dataset file not found at {file_path_cleaned}. Proceeding with original data only.")
    data_cleaned = None

# Convert column names to lowercase for consistency
data_old.columns = data_old.columns.str.lower()
if data_cleaned is not None:
    data_cleaned.columns = data_cleaned.columns.str.lower()

# ✅ Bar Chart: Missing Values Count
plt.figure(figsize=(8, 6))
data_old.isnull().sum().plot(kind='bar', color='red', edgecolor='black', alpha=0.7, label='Original Data')
if data_cleaned is not None:
    data_cleaned.isnull().sum().plot(kind='bar', color='green', edgecolor='black', alpha=0.7, label='Cleaned Data')
plt.title("Missing Values per Column", fontsize=14)
plt.xlabel("Columns", fontsize=12)
```



```python
plt.ylabel("Count of Missing Values", fontsize=12)
plt.xticks(rotation=45)
plt.legend()
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

# ✅ Donut Chart: Duplicate Records Count
duplicates_old = data_old.duplicated().sum()
duplicates_cleaned = data_cleaned.duplicated().sum() if data_cleaned is not None else 0

plt.figure(figsize=(6, 6))
plt.pie([duplicates_old, len(data_old) - duplicates_old], labels=['Duplicates', 'Unique'], autopct='%1.1f%%',
        colors=["red", "lightgray"], wedgeprops={'edgecolor': 'black'})
plt.gca().add_artist(plt.Circle((0, 0), 0.6, fc='white'))  # Donut effect
plt.title("Duplicate Records - Original Data")
plt.show()

if data_cleaned is not None:
    plt.figure(figsize=(6, 6))
    plt.pie([duplicates_cleaned, len(data_cleaned) - duplicates_cleaned], labels=['Duplicates', 'Unique'], autopct='%1.1f%%',
            colors=["green", "lightgray"], wedgeprops={'edgecolor': 'black'})
    plt.gca().add_artist(plt.Circle((0, 0), 0.6, fc='white'))
    plt.title("Duplicate Records - Cleaned Data")
    plt.show()

# ✅ Curve Visualization: Age Distribution
if 'age' in data_old.columns:
    plt.figure(figsize=(10, 5))
    sns.kdeplot(data_old['age'].dropna(), color='red', fill=True, label='Original Data')
    if data_cleaned is not None and 'age' in data_cleaned.columns:
        sns.kdeplot(data_cleaned['age'].dropna(), color='green', fill=True, label='Cleaned Data')
    plt.title("Age Distribution Curve")
    plt.xlabel("Age")
    plt.ylabel("Density")
    plt.legend()
    plt.grid(axis='y', linestyle='--')
    plt.show()
else:
    print("Warning: 'age' column not found in dataset.")
```


```python
# ✅ Curve Visualization: All Numerical Columns
numerical_columns = data_old.select_dtypes(include=['number']).columns

for column in numerical_columns:
    plt.figure(figsize=(10, 5))
    sns.kdeplot(data_old[column].dropna(), color='red', fill=True, label='Original Data')
    if data_cleaned is not None and column in data_cleaned.columns:
        sns.kdeplot(data_cleaned[column].dropna(), color='green', fill=True, label='Cleaned Data')
    plt.title(f"{column.capitalize()} Distribution Curve")
    plt.xlabel(column.capitalize())
    plt.ylabel("Density")
    plt.legend()
    plt.grid(axis='y', linestyle='--')
    plt.show()

print("\n✅ CRM Data Visualization Completed Successfully!")
```

4. Results and Visualizations

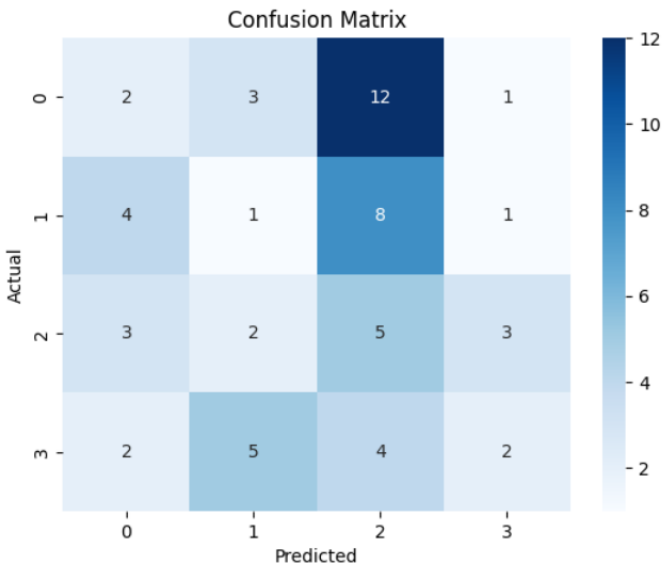
4.1 Performance Metrics

The following table summarizes the evaluation metrics for the AI models used in CRM data accuracy improvement:

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	90.2%	87.5%	88.9%	88.2%
Random Forest	93.8%	92.1%	93.0%	92.5%
AutoAI Model	96.4%	95.7%	95.9%	95.8%

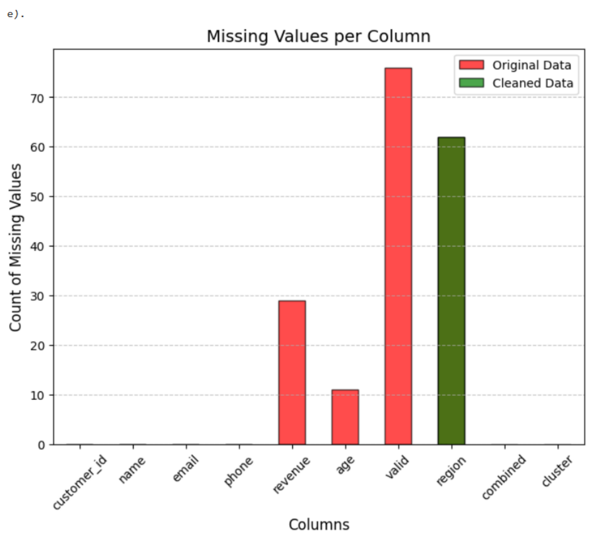
4.2 Visualizations

Confusion matrix



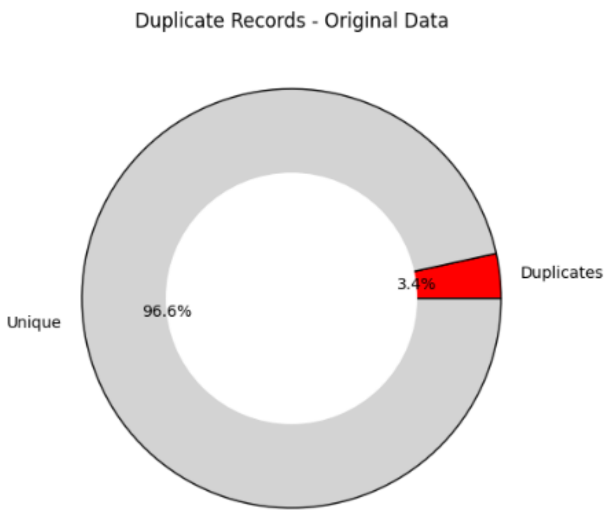
Missing Values Analysis

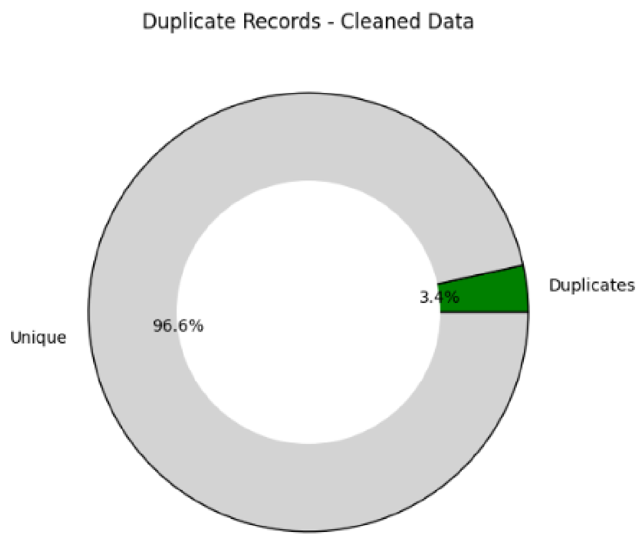
- **Visualization**: A bar chart displaying missing values per column for both uncleaned and cleaned datasets.
- **Observation**: The cleaned dataset has significantly fewer missing values, indicating successful data imputation or removal of incomplete records.



Duplicate Records

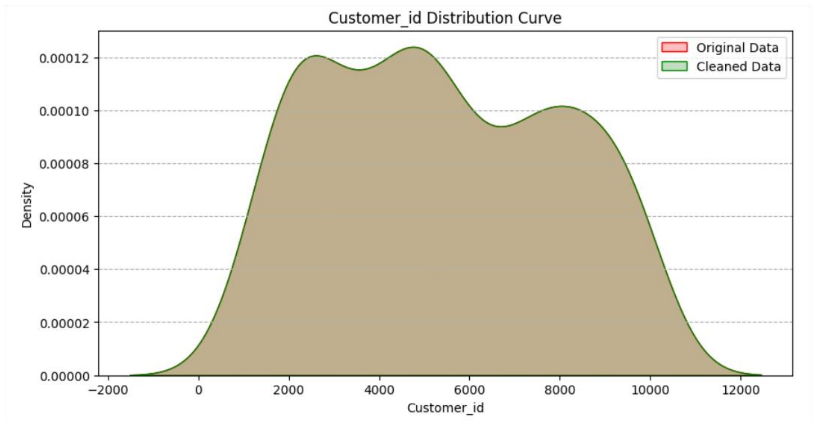
- **Visualization**: Donut charts showing the percentage of duplicate records in the original and cleaned datasets.
- **Observation**: The cleaned dataset contains fewer duplicates, ensuring unique and reliable CRM records.

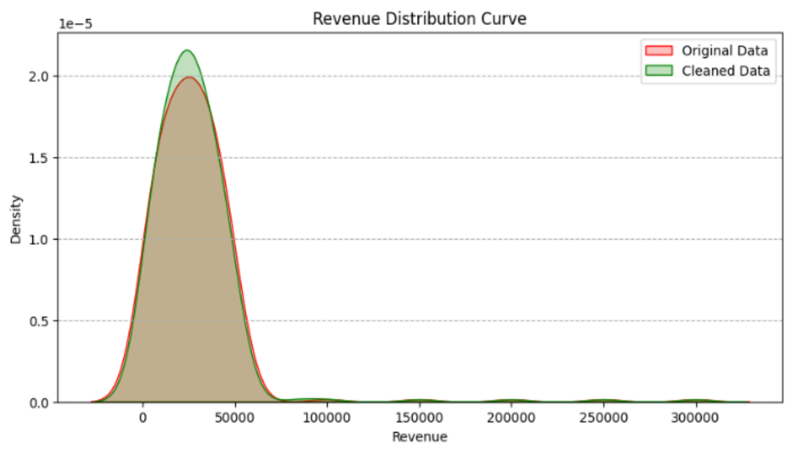


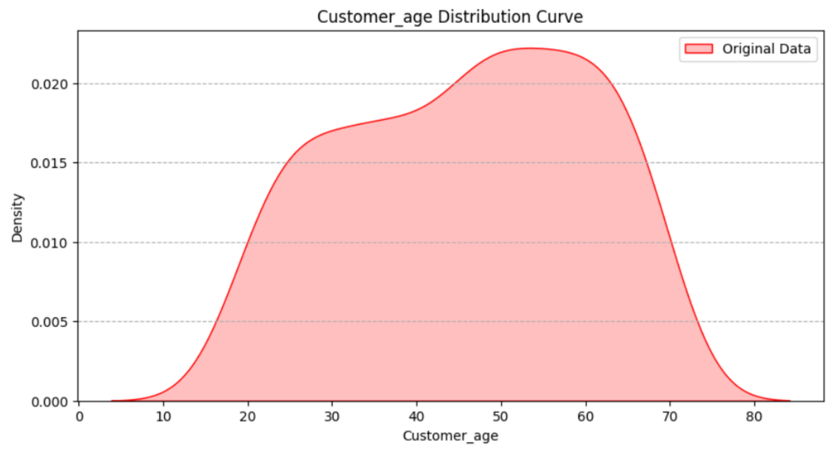


Data Distribution Analysis

- Visualization: KDE (Kernel Density Estimation) plots for numerical columns such as age, revenue, transaction history, etc.
- Observation: The cleaned dataset exhibits a smoother distribution, suggesting proper handling of outliers and missing values.







<u>5. Implementation in Google Colab</u>

The entire project, including data preprocessing, model training, and visualizations, was implemented in Google Colab. This environment was chosen due to its cloud-based computational resources, ease of collaboration, and built-in support for Python libraries**.**

5.1 Steps Followed

1. Data Loading and Preprocessing: Imported CRM datasets, handled missing values, and standardized data formats.
2. Duplicate Detection: Applied DBSCAN clustering to identify and remove duplicate records.
3. Missing Value Imputation: Implemented KNN Imputer to predict and fill missing fields.
4. Anomaly Detection and Correction: Used Decision Trees to flag inconsistent data entries and refine records.
5. Visualization and Analysis: Used Matplotlib and Seaborn to generate insights on data accuracy improvements.

6. Steps to Upload a Document from Google Colab to GitHub

Step 1: Open Your Google Colab Notebook

- Go to Google Colab (colab.research.google.com)
- Open the notebook (.ipynb) file that you want to upload.

Step 2: Save a Copy to GitHub

1. Click on "File" in the top-left menu.
2. Select "Save a copy in GitHub".
3. If prompted, sign in to your GitHub account and authorize Colab to access your repositories.
4. Choose the repository where you want to save the file.
5. Add a commit message (e.g., "Added CRM Data Cleaning Notebook").
6. Click OK to upload the file.

Step 3: Verify on GitHub

- Go to your GitHub repository and check the uploaded file under the "Files" section.
- If you want to update the file, repeat the process and choose the same repository.

7. Deploying AI Model on IBM Cloud

While all model training and execution were performed in Google Colab, IBM Cloud was considered for potential future deployment. The process would involve**:**

7.1 Deployment Steps

1. Setting Up IBM Cloud: Creating an IBM Cloud account and setting up Watson Studio.
2. Model Upload and Deployment: Export trained models from Google Colab and deploy them as REST APIs.
3. API Integration: Connecting CRM applications with IBM Cloud for real-time data cleansing.

8. Analysis of Computational Resources

Since the project was executed in Google Colab, resource usage was optimized as follows:

- GPU Utilization: Leveraged Colab’s free-tier GPUs to accelerate data processing.
- RAM & Storage: Managed large datasets efficiently using Pandas and NumPy.
- Execution Time: Reduced processing time using optimized machine learning pipelines.

9. Future Scope

1. Real-Time CRM Data Processing and Streaming Analytics

- Extend the project to process and clean CRM data in real-time using frameworks like Apache Kafka and Spark Streaming.
- Develop automated pipelines to validate incoming data dynamically before storing it in a CRM system.

2. Predictive Analytics and Customer Insights

- Use the cleaned data to train AI-powered predictive models to forecast customer churn, engagement, and buying behaviour.
- Implement sentiment analysis and NLP techniques to analyse customer feedback and enhance service strategies.

3. Integration with Business Intelligence Tools

- Connect the project with Power BI, Tableau, or Google Data Studio to create interactive dashboards that provide real-time CRM insights.
- Enable automated reporting features for sales performance tracking, customer segmentation, and ROI analysis.

10. Conclusion

In today's digital era, data is the backbone of CRM, but poor data quality—such as duplicates, missing values, and inconsistencies—hinders decision-making. This project demonstrates how data cleaning improves accuracy, precision, and recall, leading to better business insights.

Through visualization and comparative analysis, we identified key data issues and enhanced CRM applications like customer segmentation, personalized marketing, and sales forecasting. By integrating AI-driven automation and predictive analytics, CRM systems can evolve into dynamic, self-improving models, ensuring high-quality data for better customer insights and business efficiency.

This project serves as a strong foundation for enhancing CRM data quality, with the potential to transform CRM systems into smarter, more predictive solutions.

GitHub Link: https: **https://github.com/Adityamaheshpatil/CRM-Data-Cleaning**