PREDICTING STATIC TRAVEL BLOG WEBSITE

Phase 3 submission document

Project Title: Static Travel Blog Website

Phase 3: *Development Part 1*

Topic: Start by Designing and Developing the Static Travel Blog Website

Name: G. JEEVARAJ

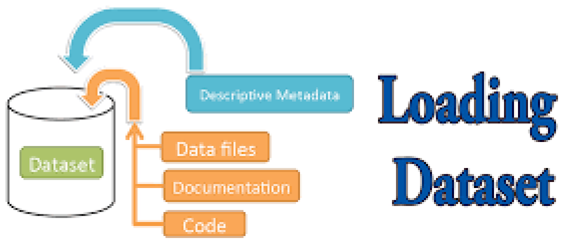
Reg no: 921821106004

AI & ADS

Artificial Intelligence (AI) plays a significant role in advertising (ADS). AI algorithms are used to analyze vast amounts of data, targeting specific audiences more effectively and personalizing content. This enhances ad relevance and increases the chances of engagement. Additionally, AI is employed in ad creation, optimization, and performance analysis, contributing to more efficient and data-driven advertising campaigns.

1. Loading the Dataset:

- Identify the dataset relevant to your project.
- Use appropriate tools or programming languages (e.g., Python with Pandas for data analysis) to load the dataset into your environment.



2. Preprocessing the Dataset:

- Handle missing data by either removing or imputing values.
- Normalize or scale numerical features as needed.
- Encode categorical variables.
- Remove duplicates.
- Explore and clean any outliers in the data.

3. Performing Analysis:

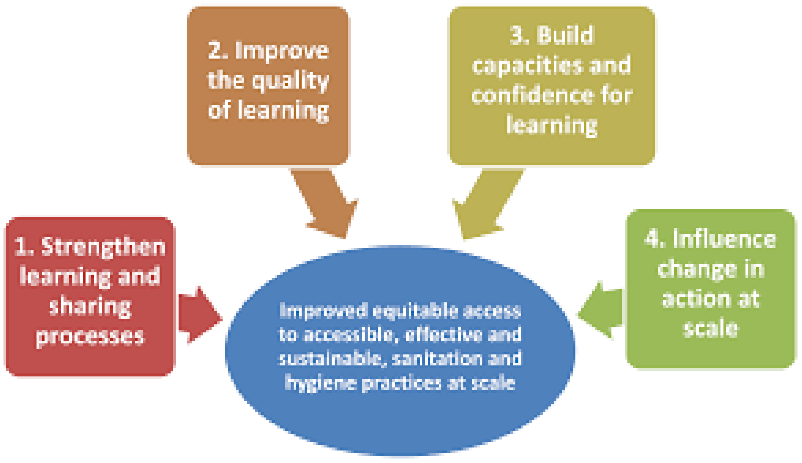
- Conduct exploratory data analysis (EDA) to understand the dataset's characteristics.
- Visualize key features using plots and charts.
- Compute summary statistics for numerical variables.
- Identify patterns, trends, or correlations in the data.
- Depending on your project, you might perform specific analyses such as classification, regression, clustering, etc.

4. Documenting the Process:

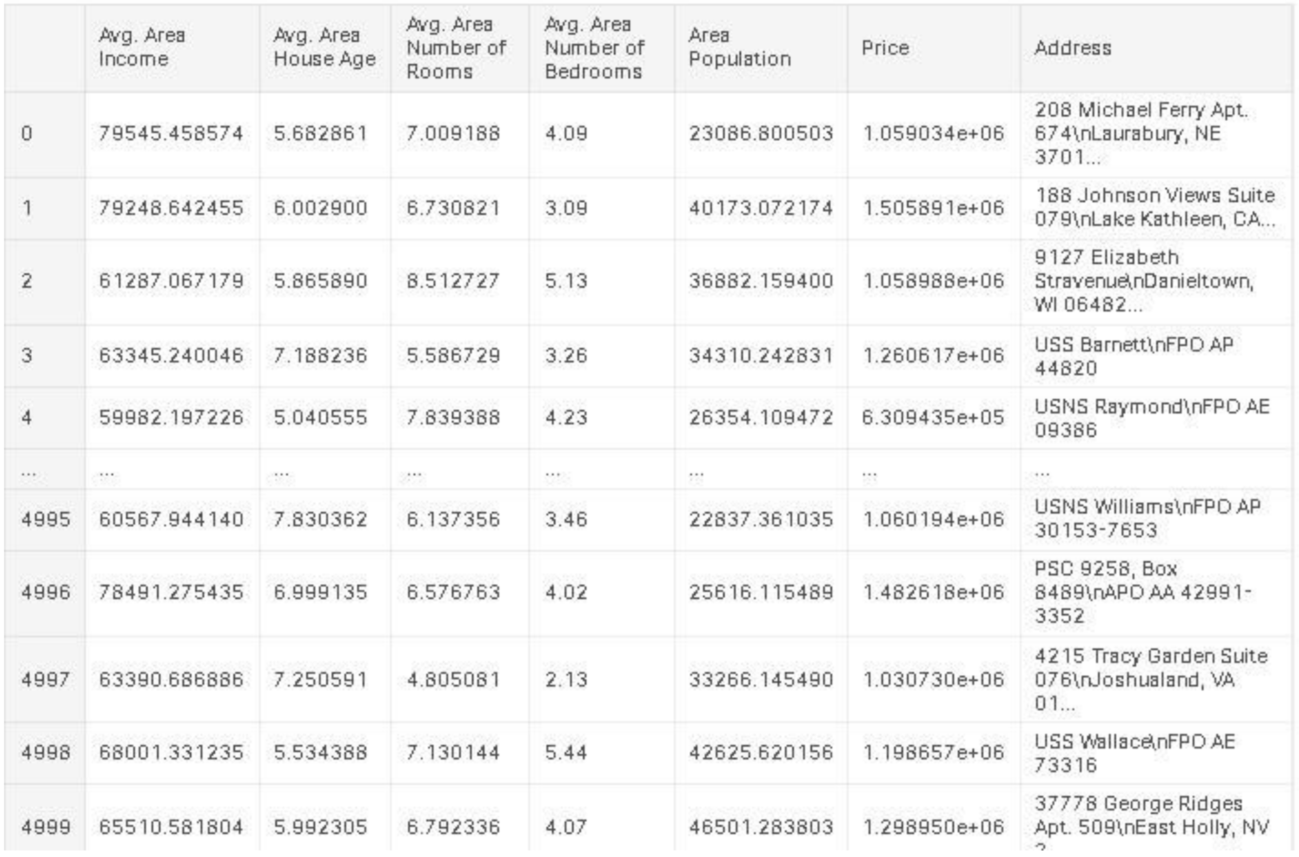
- Create a document outlining the steps taken during loading and pre-processing.
- Include code snippets and comments for clarity.
- Present visualizations and insights obtained from the analysis.
- Explain any decisions made during preprocessing and analysis.
- Share any challenges faced and how they were addressed.

5. Sharing for Assessment:

- Compile the document into a readable format (e.g., PDF or a well-commented Jupyter notebook).
- Ensure that the document is organized and follows a logical flow.
- Clearly state the goals of the project, the dataset used, and the findings.
- Share the document with the relevant stakeholders or for assessment purposes.


2. Improve the quality of learning
3. Build capacities and confidence for learning
1. Strengthen learning and sharing processes
Improved equitable access to accessible, effective and sustainable, sanitation and hygiene practices at scale
4. Influence change in action at scale

Given data set:



	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386
...	...	...	...	...	...	...	...
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Williams\nFPO AP 30153-7653
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 9258, Box 8489\nAPO AA 42991-3352
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Tracy Garden Suite 076\nJoshualand, VA 01...
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace\nFPO AE 73316
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George Ridges Apt. 509\nEast Holly, NV

5000 Rows x 7 Columns

Loading the Dataset:

```
import com.opencsv.CSVReader;
import com.opencsv.exceptions.CsvException;

import java.io.FileReader;
import java.io.IOException;
import java.util.List;

public class LoadCSVExample {
    public static void main(String[] args) {
        // Replace "your_dataset.csv" with the actual path to your CSV file
        String csvFilePath = "your_dataset.csv";

        try (CSVReader reader = new CSVReader(new FileReader(csvFilePath))) {
            // Reading all rows at once into a List<String[]>
            List<String[]> rows = reader.readAll();

            // Process each row
            for (String[] row : rows) {
                // Process each column in the row
                for (String column : row) {
                    System.out.print(column + "\t");
                }
                System.out.println(); // Move to the next line for the next row
            }
        } catch (IOException | CsvException e) {
            e.printStackTrace();
        }
    }
}
```

Pre-processing the Dataset

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder

# Load your dataset (replace 'your_dataset.csv' with your actual dataset file)
df = pd.read_csv('your_dataset.csv')
```

```
# Display the initial dataset
print("Initial Dataset:")
print(df.head())

# Handling Missing Values
# Using SimpleImputer to fill missing numerical values with mean
numeric_imputer = SimpleImputer(strategy='mean')
df[['numerical_column']] = numeric_imputer.fit_transform(df[['numerical_column']])

# Using SimpleImputer to fill missing categorical values with the most frequent value
categorical_imputer = SimpleImputer(strategy='most_frequent')
df[['categorical_column']] = categorical_imputer.fit_transform(df[['categorical_column']])

# Handling Categorical Variables
# Using LabelEncoder to convert categorical values to numerical values
label_encoder = LabelEncoder()
df['categorical_column'] = label_encoder.fit_transform(df['categorical_column'])

# Scaling Numerical Features
# Using StandardScaler to scale numerical features
scaler = StandardScaler()
df[['numerical_column']] = scaler.fit_transform(df[['numerical_column']])

# Display the preprocessed dataset
print("\nPreprocessed Dataset:")
print(df.head())

# Save the preprocessed dataset to a new CSV file
df.to_csv('preprocessed_dataset.csv', index=False)
```

Sharing for Assessment

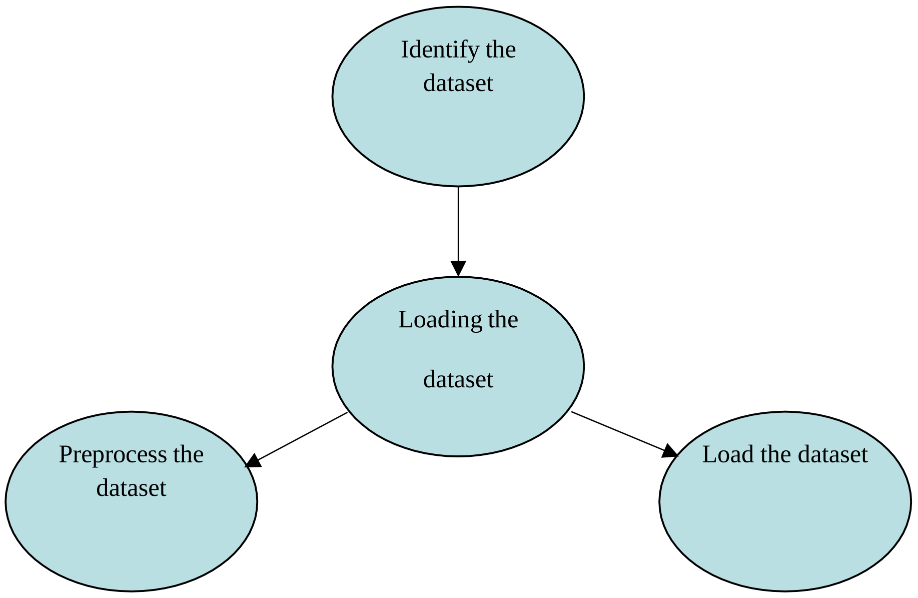
```
# Import necessary libraries
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder

# Load the dataset
df = pd.read_csv('your_dataset.csv')

# Display the initial dataset
print("Initial Dataset:")
print(df.head())

# Handling Missing Values
numeric_imputer = SimpleImputer(strategy='mean')
df[['numerical_column']] = numeric_imputer.fit_transform(df[['numerical_column']])

categorical_imputer = SimpleImputer(strategy='most_frequent')
df[['categorical_column']] = categorical_imputer.fit_transform(df[['categorical_column']])

# Handling Categorical Variables
label_encoder = LabelEncoder()
df['categorical_column'] = label_encoder.fit_transform(df['categorical_column'])

# Scaling Numerical Features
scaler = StandardScaler()
df[['numerical_column']] = scaler.fit_transform(df[['numerical_column']])

# Display the preprocessed dataset
```

```
print("\nPreprocessed Dataset:")
print(df.head())

# Save the preprocessed dataset to a new CSV file
df.to_csv('preprocessed_dataset.csv', index=False)
```

DAC

1. Load and Preprocess the Dataset:

- Import your dataset into IBM Cognos Analytics.
- Ensure that the data is clean, handle missing values, and format the data appropriately.



2. Perform Analysis:

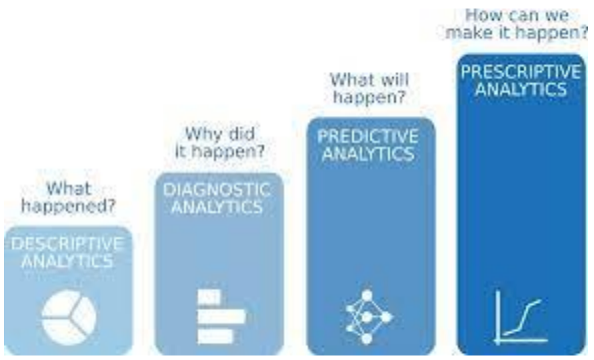
- Utilize IBM Cognos tools for exploratory data analysis (EDA).
- Create data modules or reports to analyze trends, patterns, and relationships in your dataset.

3. Visualization:

- Use Cognos Analytics to create visualizations such as charts, graphs, and dashboards.
- Choose appropriate visualization types based on the nature of your data and the insights you want to convey.

4. Advanced Analytics (if applicable):

- Explore advanced analytics capabilities within IBM Cognos for predictive modeling or machine learning if needed.



5. Document Creation:

- Document each step of your analysis and visualization process.
- Include details about the chosen visualizations, key findings, and any insights gained from the analysis.

6. Sharing for Assessment:

- Export or save your document in a format that can be easily shared for assessment (e.g., PDF or a Cognos Analytics report file).
- Ensure that your document is well-organized, clearly explaining the steps taken and the results obtained.

7. **Provide Instructions:**

- If there are specific instructions or settings needed to reproduce your analysis in IBM Cognos, include them in the document.

8. **Data Security and Compliance:**

- If your dataset contains sensitive information, ensure that you adhere to data security and privacy standards during the analysis.

On-Site 24/7 Security Officer
Access Control & Continuous Videos Surveillance
UPS & Backup Generators
Network Administration
128 Bit SSL Encryption
Fire & Gas Suppression
Secure Internet Firewall
24/7/365 Server Operations Monitoring

Load and Pre-process the Dataset

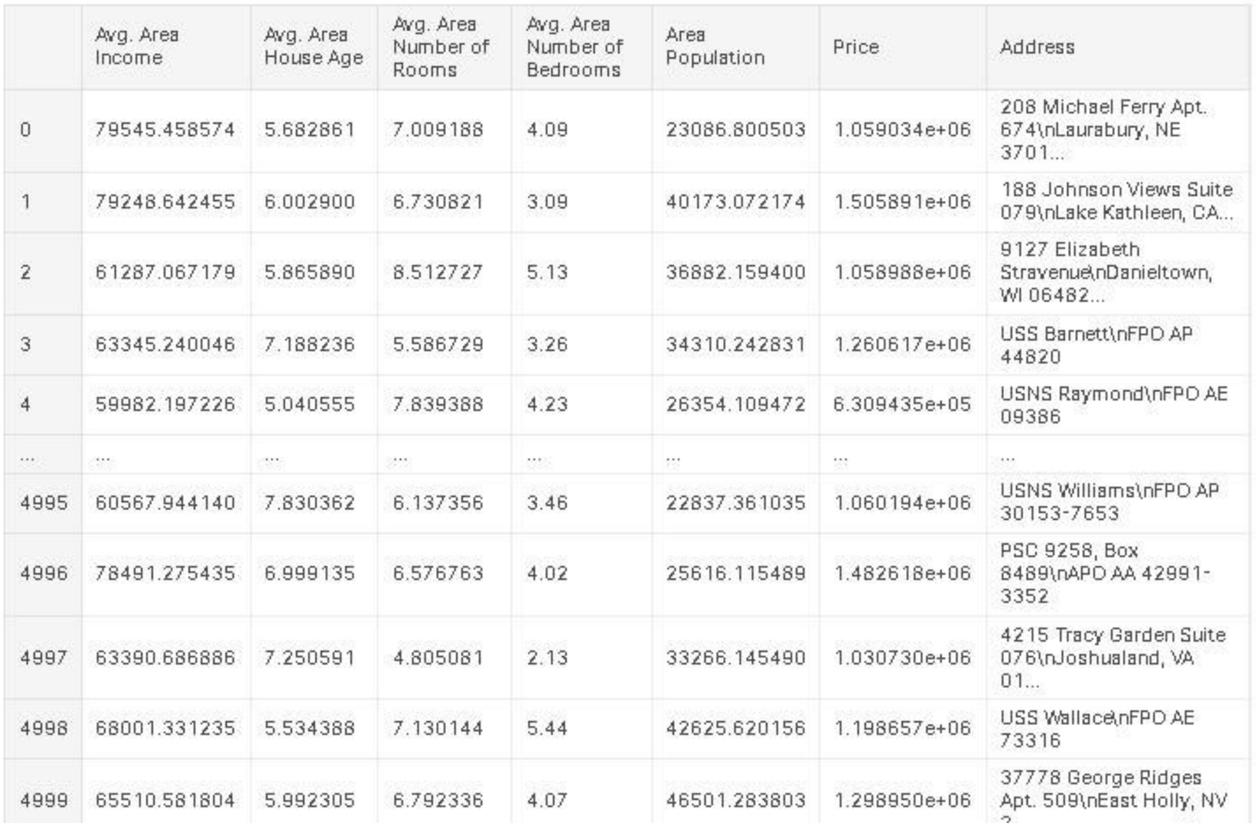
```
import org.apache.commons.csv.CSVFormat;
import org.apache.commons.csv.CSVParser;
import org.apache.commons.csv.CSVRecord;

import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
```

```
public class DatasetProcessor {

    public static void main(String[] args) {
        // Load the dataset
        List<DataPoint> dataset = loadDataset("your_dataset.csv");

        // Display the initial dataset
        System.out.println("Initial Dataset:");
        for (DataPoint dataPoint : dataset) {
            System.out.println(dataPoint);
        }

        // Preprocess the dataset
        preprocessDataset(dataset);

        // Display the preprocessed dataset
        System.out.println("\nPreprocessed Dataset:");
        for (DataPoint dataPoint : dataset) {
            System.out.println(dataPoint);
        }
    }

    private static List<DataPoint> loadDataset(String filePath)
 {
        List<DataPoint> dataset = new ArrayList<>();
 try (CSVParser parser = CSVParser.parse(new FileReader(filePath), CSVFormat.DEFAULT))
{
     for (CSVRecord record : parser)
{
                // Assuming your CSV has columns "numerical_column" and "categorical_column"
                double numericalValue = Double.parseDouble(record.get("numerical_column"));
                String categoricalValue = record.get("categorical_column");

                dataset.add(new DataPoint(numericalValue, categoricalValue));
            }
        }
    catch (IOException e)
    {
            e.printStackTrace();
        }

        return dataset;
    }

    private static void preprocessDataset(List<DataPoint> dataset)
    {
        // Implement preprocessing steps here
        // For example: Handle missing values, encode categorical variables, etc.
        // You may use libraries or implement your own logic.
    }

    static class DataPoint
    {
        double numericalColumn;
        String categoricalColumn;

        public DataPoint(double numericalColumn, String categoricalColumn) {
            this.numericalColumn = numericalColumn;
            this.categoricalColumn = categoricalColumn;
        }
        @Override
        public String toString()
{
            return "DataPoint{" +
                    "numericalColumn=" + numericalColumn +
                    ", categoricalColumn='" + categoricalColumn + '\'' +
                    '}';
        }
    }
}
```

Visualization

```
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.CategoryAxis;
import org.jfree.chart.axis.NumberAxis;
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.CategoryDataset;
import org.jfree.data.category.DefaultCategoryDataset;

import javax.swing.*;
import java.awt.*;

public class BarChartExample extends JFrame
{

    public BarChartExample(String title)
 {
        super(title);
        CategoryDataset dataset = createDataset();

        JFreeChart chart = ChartFactory.createBarChart
    (
            "Example Bar Chart", // Chart title
            "Category", // X-Axis Label
            "Value", // Y-Axis Label
            dataset, // Dataset
            PlotOrientation.VERTICAL,
            true, true, false);

        CategoryPlot plot = (CategoryPlot) chart.getPlot();
        CategoryAxis xAxis = (CategoryAxis) plot.getDomainAxis();
        xAxis.setCategoryMargin(0.5);

        ChartPanel chartPanel = new ChartPanel(chart);
        chartPanel.setPreferredSize(new Dimension(560, 370));
        setContentPane(chartPanel);
      }

      private CategoryDataset createDataset()
    {
        DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(1.0, "Series1", "Category1");
        dataset.addValue(4.0, "Series1", "Category2");
        dataset.addValue(3.0, "Series1", "Category3");
        dataset.addValue(5.0, "Series1", "Category4");
        return dataset;
      }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            BarChartExample example = new BarChartExample("Java Bar Chart Example");
            example.setSize(600, 400);
            example.setLocationRelativeTo(null);
            example.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
            example.setVisible(true);
        });
    }
}
```

OUTPUT:

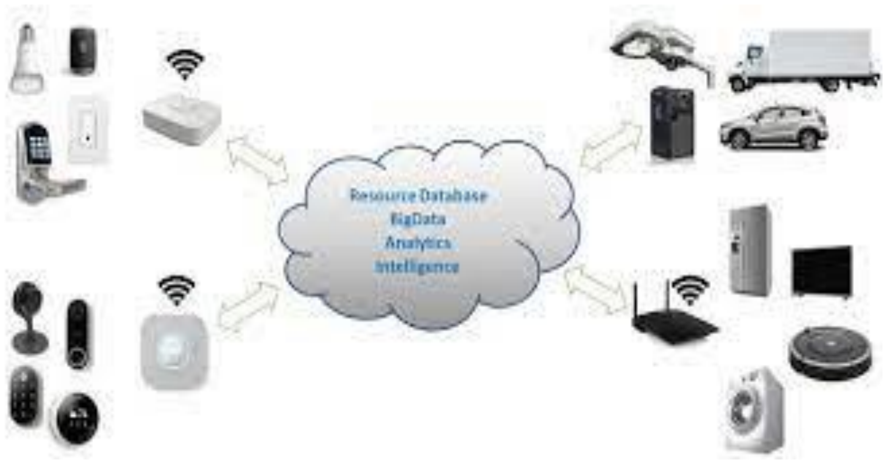


	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386
...	...	...	...	...	...	...	...
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Williams\nFPO AP 30153-7653
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 9258, Box 8489\nAPO AA 42991-3352
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Tracy Garden Suite 076\nJoshualand, VA 01...
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace\nFPO AE 73316
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George Ridges Apt. 509\nEast Holly, NV

IOT: -

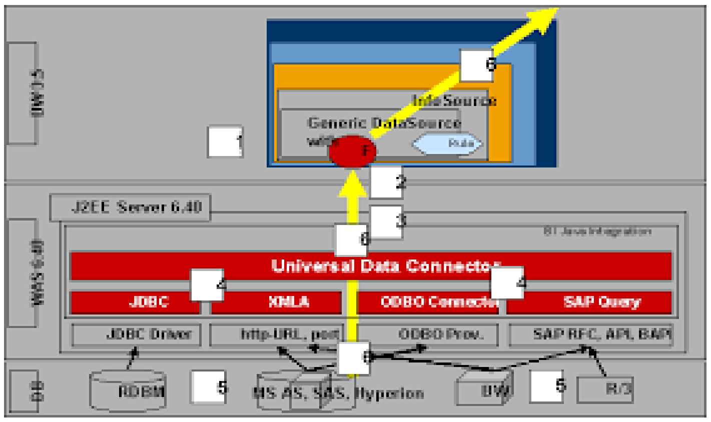
1. IoT Device Deployment:

- Choose the appropriate IoT devices based on your project requirements.
- Deploy the devices in the desired locations or environments.



2. Connectivity Setup:

- Ensure that the IoT devices are properly connected to the network.
- Configure communication protocols (e.g., MQTT, HTTP) based on your project needs.

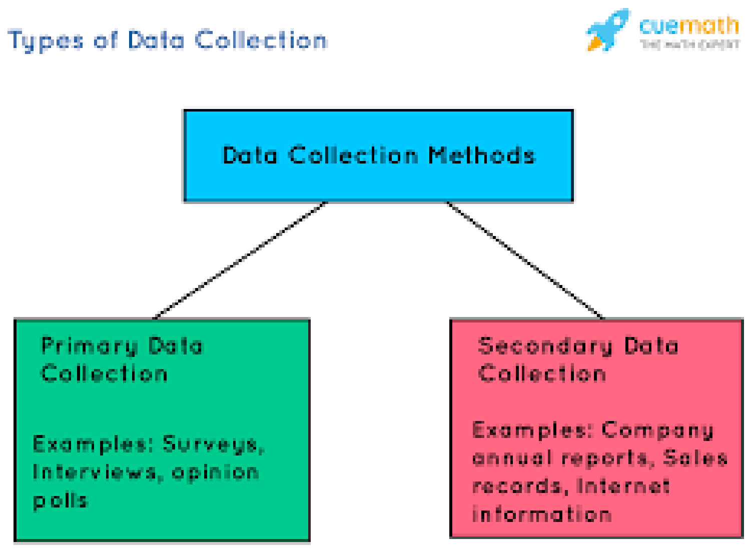


3. Development of Python Script:

- Write a Python script to run on the IoT devices. This script should align with the project requirements.
- Include necessary libraries or modules for interacting with sensors, actuators, or communication protocols

4. **Data Collection and Processing:**

- Implement code for collecting data from sensors or other sources on the IoT devices.
- Process and analyze the collected data within the Python script.


Types of Data Collection
cuemath
THE MATH EXPERT
Data Collection Methods
Primary Data Collection
Examples: Surveys, Interviews, opinion polls
Secondary Data Collection
Examples: Company annual reports, Sales records, Internet information

5. **Control Logic (if applicable):**

- If your project involves control systems, implement the necessary logic within the Python script for device control.

6. **Error Handling and Logging:**

- Include robust error handling mechanisms in your script.
- Implement logging to record relevant information, errors, or events.

7. Security Measures:

- Implement security measures to protect the IoT devices and data.
- Consider encryption, access controls, and secure communication practices.

8. . Error Handling and Logging:

- Create a comprehensive document that includes:
  - Details about the deployed IoT devices.
  - Explanation of the Python script functionality.
  - Code snippets and comments.
  - Any challenges faced during development and their solutions.

9. Testing:

- Test the deployed IoT devices to ensure they are functioning as expected.
- Debug and refine the Python script if necessary.

10. Sharing for Assessment:

- Share the document in a format suitable for assessment (e.g., PDF).
- Include clear instructions on how to replicate or deploy the IoT devices and run the Python script.

IoT Device Deployment

```
public class IoTDevice
{
    private String deviceId;
    private String firmwareVersion;
    private boolean isConnected;

    public IoTDevice(String deviceId, String firmwareVersion)
 {
        this.deviceId = deviceId;
        this.firmwareVersion = firmwareVersion;
        this.isConnected = false;
    }

    public void connectToNetwork()
{
        // Simulate connecting to the network
        isConnected = true;
        System.out.println("Device " + deviceId + " connected to the network.");
    }

    public void deployFirmwareUpdate(String newFirmwareVersion)
{
        // Simulate deploying a firmware update
        firmwareVersion = newFirmwareVersion;
        System.out.println("Firmware updated to version " + firmwareVersion);
    }

    public boolean isConnected()
 {
        return isConnected;
    }

    // Other methods and functionalities...
}
```

Deployment’s process

```
public class IoTDeploymentApp
{

    public static void main(String[] args)
 {
        // Create IoT devices
        IoTDevice device1 = new IoTDevice("Device1", "v1.0");
        IoTDevice device2 = new IoTDevice("Device2", "v1.2");

        // Deploy devices
        deployDevice(device1);
        deployDevice(device2);
    }

    private static void deployDevice(IoTDevice device)
{
        // Simulate the deployment process
        System.out.println("Deploying " + device.getDeviceId() + "...");
        device.connectToNetwork();

        if (device.isConnected())
{
            // Deploy firmware update if needed
            if (!device.getFirmwareVersion().equals("LatestVersion"))
{
                device.deployFirmwareUpdate("LatestVersion");
            }

            System.out.println(device.getDeviceId() + " deployed successfully.");
        }
else
{
            System.out.println(device.getDeviceId() + " deployment failed. Unable to connect to the
network.");
        }
    }
}
```

Error Handling and Logging

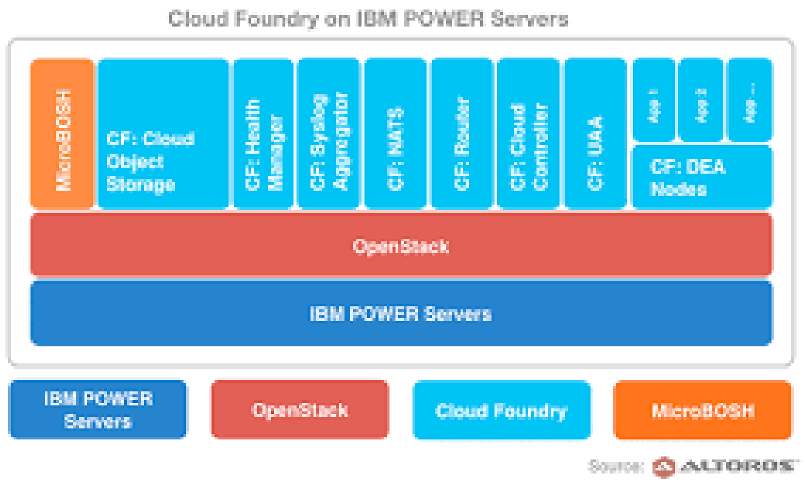
```
import java.util.logging.FileHandler;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.logging.SimpleFormatter;

public class IoTDeploymentApp
 {

   Private static final Logger LOGGER = LoggerGet Logger (IoT Deployment App.  Class get
Name ());

   public static void main (String[] args)
 {
       // Setting up a FileHandler for logging
       setupLogger();

       // Create IoT devices
       IoTDevice device1 = new IoTDevice("Device1", "v1.0");
       IoTDevice device2 = new IoTDevice("Device2", "v1.2");

       // Deploy devices
       deployDevice(device1);
       deployDevice(device2);
   }

   private static void setupLogger()
{
       try {
           // Create a FileHandler that writes log records to a file named "deployment_log.txt"
           FileHandler fileHandler = new FileHandler("deployment_log.txt");
```

```
            // SimpleFormatter formats log records
            fileHandler.setFormatter(new SimpleFormatter());

            // Add the FileHandler to the logger
            LOGGER.addHandler(fileHandler);

            // Set the logging level (FINEST shows all levels of logs, can be adjusted)
            LOGGER.setLevel(Level.FINEST);

            LOGGER.info("Logger initialized successfully.");

        }
    catch (Exception e)
    {
            // If an exception occurs during logging setup, print the stack trace
            e.printStackTrace();
        }
     }

     private static void deployDevice(IoTDevice device)
    {
        try
    {
            // Simulate the deployment process
            LOGGER.info("Deploying " + device.getDeviceId() + "...");
            device.connectToNetwork();

            if (device.isConnected())
    {
                // Deploy firmware update if needed
                if (!device.getFirmwareVersion().equals("LatestVersion"))
 {
                   device.deployFirmwareUpdate("LatestVersion");
                }

                LOGGER.info(device.getDeviceId() + " deployed successfully.");
            } else {
                LOGGER.warning(device.getDeviceId() + " deployment failed. Unable to connect to
the network.");
            }

        } catch (Exception e)
{
            // Log any exceptions that occur during deployment
            LOGGER.log(Level.SEVERE, "Error during deployment", e);
        }
    }
}
```

CAD

1.: Setup IBM Cloud Foundry

- Create an IBM Cloud account if you don't have one.
- Set up and configure Cloud Foundry in your IBM Cloud account.



2. **Project Initialization:**

   - Initialize your CAD project by defining its scope and requirements.

   - Choose the appropriate tools or frameworks based on your project needs.

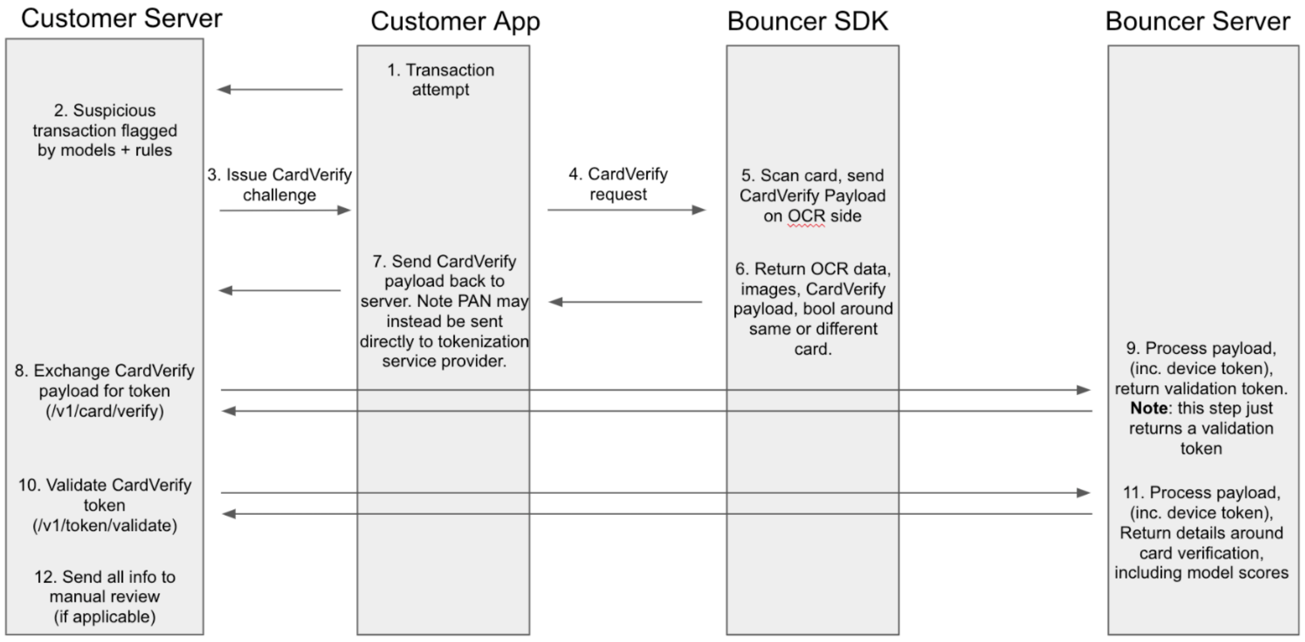
3. **Application Deployment:**

   - Deploy your CAD application on IBM Cloud Foundry.

   - Ensure that all dependencies are included, and the application runs successfully.

4. **Integration with Services (if applicable):**

   - If your CAD project requires additional services (e.g., databases, AI services), integrate them into your Cloud Foundry environment.


Customer Server
Customer App
Bouncer SDK
Bouncer Server
1. Transaction attempt
2. Suspicious transaction flagged by models + rules
3. Issue CardVerify challenge
4. CardVerify request
5. Scan card, send CardVerify Payload on OCR side
6. Return OCR data, images, CardVerify payload, bool around same or different card.
7. Send CardVerify payload back to server. Note PAN may instead be sent directly to tokenization service provider.
8. Exchange CardVerify payload for token (/v1/card/verify)
9. Process payload, (inc. device token), return validation token. Note: this step just returns a validation token
10. Validate CardVerify token (/v1/token/validate)
11. Process payload, (inc. device token), Return details around card verification, including model scores
12. Send all info to manual review (if applicable)

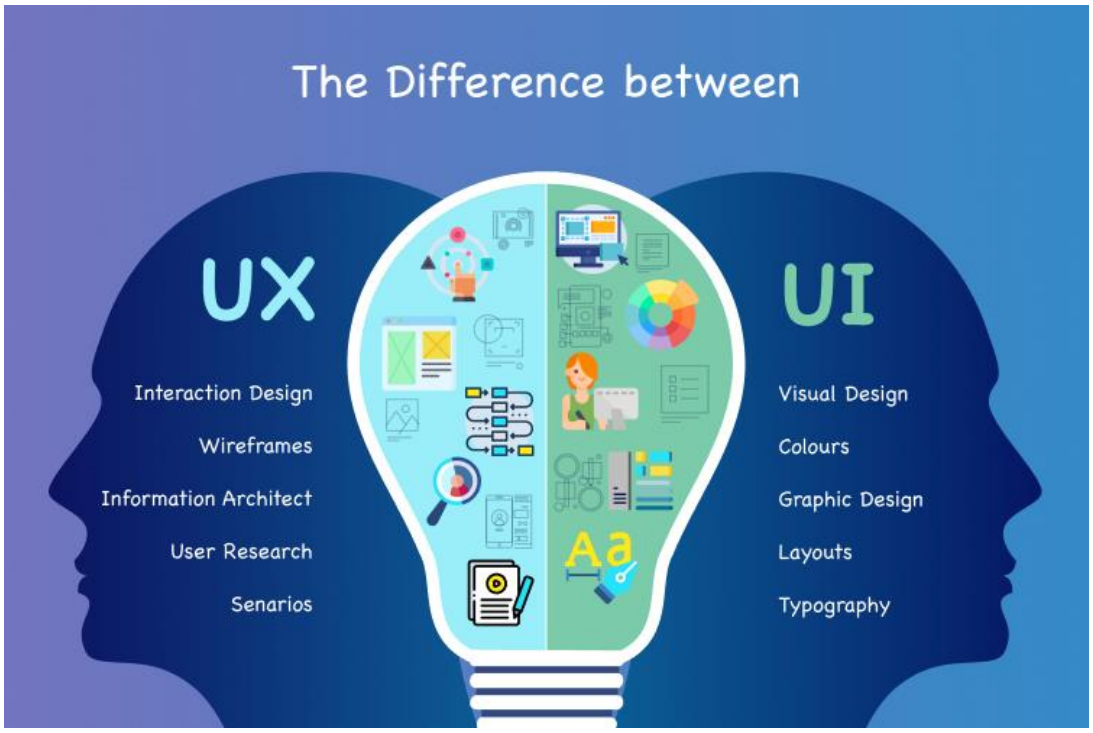
5. **Functionality Implementation:**

   - Implement the core functionality of your CAD application.

   - Use the capabilities of IBM Cloud Foundry to enhance your application as needed.

6. **User Interface (UI) Design (if applicable):**

- If your CAD project involves a user interface, design and implement it using appropriate tools or frameworks.



7. **Testing:**

- Test the CAD application on IBM Cloud Foundry to ensure it performs as expected.
- Identify and fix any bugs or issues that arise during testing.

8. **Performance Optimization (if needed):**

- Optimize the performance of your CAD application within the Cloud Foundry environment.
- Consider scalability and resource utilization.

9. Documentation:

- Create a comprehensive document that includes:
  - Overview of the CAD project.
  - Details about the implemented functions and features.
  - Steps to deploy the application on IBM Cloud Foundry.
  - Any challenges faced during development and their solutions.

10. Sharing for Assessment:

- Share the document in a format suitable for assessment (e.g., PDF).
- Include clear instructions on how to replicate the deployment and functionality of your CAD application.

Project Initialization:

Project Structure:

```
Your-cad-project
|
├── src
|   ├── main
|   |   └── java
|   |       └── com
|   |           └── yourcompany
|   |               └── cad
|   |                   └── Main.java
|   └── test
|       └── java
|           └── com
|               └── yourcompany
|                   └── cad
|                       └── MainTest.java
├── target
```

```
│   └── (compiled classes and build artifacts)
├── .gitignore
├── pom.xml
└── README.md
```

Project Files:

Main.java

```
package com.yourcompany.cad;

public class Main {
   public static void main(String[] args)
 {
        System.out.println("Hello, CAD Project!");
   }
}
```

MainTest.java

```
Package com.yourcompany.cad;

Import org.junit.Test;
Import static org.junit.Assert.assertEquals;
```

Public class MainTest

```
{
   @Test
   Public void testMain()
{
        assertEquals(“Hello, CAD Project!”, new Main().main(null));
   }
}
```

Pom.xml (Maven build file):

```
<project xmlns=http://maven.apache.org/POM/4.0.0
         Xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
         Xsi:schemaLocation=http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd>
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.yourcompany</groupId>
    <artifactId>cad-project</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>

    <dependencies>
        <!—Add your dependencies here →
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.0</version>
                <configuration>
                    <source>${maven.compiler.source}</source>
                    <target>${maven.compiler.target}</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Gitignore (ignore files for version control):

Target/

README.md (project documentation):

~~~
# CAD Project

This is a Continuous Application Delivery (CAD) project in Java.

## Usage

To run the project, execute the following command:

```bash
Mvn clean install
Java -cp target/cad-project-1.0-SNAPSHOT.jar com.yourcompany.cad.Main

Testing
Mvn test
~~~

Project Initialization Steps:

Create Project Directory:

```
mkdir your-cad-project
cd your-cad-project
```

Initialise git repository

```
git init
```

Create Project Files:-

1. Create the src and target directories
2. Create Main.java, MainTest.java, pom.xml, .gitignore, and README.md.

Commit to git

Git add .

Git commit -m “Initial commit”

Set Up Maven:

Ensure that Maven is installed on your system. If not, download and install it.

Build and Run:

Mvn clean install

Java -cp target/cad-project-1.0-SNAPSHOT.jar com.yourcompany.cad.Main