National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Geerthana Moorthy
Student ID: x23267348

School of Computing
National College of Ireland

Supervisor: Aaloka Anant

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	GEERTHANA MOORTHY
Student ID:	X23267348
Programme:	M.SC Data Analytics **Year:** 2024-2025
Module:	Research Practicum
Lecturer: **Submission Due Date:**	11-08-2025
Project Title:	Analyzing Gender Disparities in Corporate Leadership Using Machine Learning Techniques on Global Board Member Data
Word Count:	1029 **Page Count:** 5

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Geerthana Moorthy

Date: 11-08-2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Geerthana Moorthy
X23267348

1 Introduction

1.1 Project Overview

This project provides a machine learning-driven solution to analyze and predict gender representation on corporate boards. Using a comprehensive synthetic dataset of global board members, we have trained a model to identify the key factors that correlate with a board member being female.

The final output is a web service, built with Flask, that exposes this predictive model through a REST API. This allows users to submit the profile of a hypothetical board member and receive a prediction regarding their gender, along with a confidence score.

1.2 Core Functionality

- **Data Generation:** A Jupyter notebook (datageneration.ipynb) is provided to create a realistic, large-scale synthetic dataset.
- **Model Training:** A second notebook (modelling.ipynb) handles data cleaning, exploratory data analysis (EDA), feature engineering, and the training, evaluation, and selection of a machine learning model (XGBoost was chosen as the best performer).
- **Prediction Service:** The Flask application (app.py) loads the trained model and associated components to serve predictions via an API.

1.3 Technology Stack

- **Backend:** Python, Flask
- **Data Science & ML:** Pandas, NumPy, Scikit-learn, XGBoost, Faker
- **Visualization (in notebooks):** Matplotlib, Seaborn, Plotly

2 Project Structure

The repository is organized as follows:

```
/
├── models/
│   ├── final_model.pkl           # The trained XGBoost model
│   ├── pca_transformer.pkl       # The saved PCA transformer
│   ├── preprocessor.pkl          # The saved data preprocessor
│   ├── feature_names.pkl         # List of feature names used for training
│   └── performance_metrics.pkl   # Performance metrics of the final model
│
├── templates/
│   └── index.html                # The HTML frontend for the application
│
├── dataset/
│   └── dataset.csv               # The synthetic dataset (generated)
│
```

```
├── app.py                  # The main Flask application file
├── datageneration.ipynb    # Notebook to generate synthetic data
├── modelling.ipynb         # Notebook for EDA and model training
└── requirements.txt        # Python dependencies
```

3 Prerequisites

- Python (3.8 or newer)
- pip (Python package installer)
- Git
- (Optional) A tool to run Jupyter notebooks, like Jupyter Lab or VS Code.

4 Installation and Setup Guide

Follow these steps to set up and run the application on your local machine.

5 Step 1: Clone the Repository

```
git clone <your-repository-url>
cd <repository-directory>
```

6 Step 2: Set Up Virtual Environment

It is highly recommended to use a virtual environment to manage dependencies and avoid conflicts.

```
# Create a virtual environment
python -m venv venv

# Activate the virtual environment
# On Windows
venv\Scripts\activate
# On macOS/Linux
source venv/bin/activate
```

7 Step 3: Install Dependencies

Install all required Python libraries using the requirements.txt file.

```
pip install -r requirements.txt
```

8 Step 4: Data and Model Generation (Optional)

The application requires the dataset in dataset/ and model artifacts in models/. You have two options:

- **Use Pre-existing Files:** If the dataset.csv and the files within the models/ directory are already provided, you can skip this step.
- **Generate Files Manually:** If you need to generate them from scratch:
    1. **Generate Data:** Open and run all cells in the datageneration.ipynb notebook. This will create dataset/dataset.csv.
    2. **Train Model:** Open and run all cells in the modelling.ipynb notebook. This will populate the models/ directory with the necessary .pkl files.

9 Step 5: Run the Application

Once the dependencies are installed and the model files are in place, you can start the Flask server.

```
python app.py
```

You will see output similar to this, indicating the server is running:

```
All models loaded successfully!
```

Model type: XGBoost
Model accuracy: 0.7914
 * Serving Flask app 'app'
 * Running on http://0.0.0.0:5000
 * Press CTRL+C to quit
You can now access the application by navigating to http://127.0.0.1:5000 in your web browser.

10 Application Endpoints

The Flask application exposes the following endpoints:

10.1 GET /

- **Description:** The main landing page of the application.
- **Functionality:** Renders an HTML interface that displays the key performance metrics of the loaded model and provides sample JSON data that can be used for testing the prediction endpoint.

10.2 POST /predict

- **Description:** The core prediction endpoint. It accepts a JSON object describing a board member and returns a gender prediction.
- **Input:** Content-Type: application/json
- **Output:** A JSON object containing the prediction and associated metadata.

10.3 GET /api/schema

- **Description:** Provides the expected JSON schema for the /predict endpoint.
- **Functionality:** This is useful for developers to understand the required fields, data types, and acceptable values for making a prediction request.

10.4 GET /health

- **Description:** A health check endpoint to monitor the application's status.
- **Functionality:** Returns a JSON object indicating if the application is "healthy" (i.e., all model components are loaded correctly) or "unhealthy".

11 How to Use the Prediction API

11.1 Input JSON Structure

The /predict endpoint expects a JSON payload with the following structure. All fields listed in the schema (/api/schema) are required.


```
{
   "Age": 45,
   "Tenure_in_Years": 3,
   "Committee_Memberships": 2,
   "Education_Level": "Master's Degree",
   "Total_Board_Members": 12,
   "Female_Board_Percentage": 41.67,
   "Avg_Board_Age": 54.0,
   "Avg_Board_Tenure": 3.2,
   "Leadership_Positions": 2,
   "PhD_Percentage": 25.0,
   "Gender_Diversity_Score": 83.34,
```

```
    "Is_Executive": 0,
    "Is_Leadership_Role": 1,
    "Board_Role": "Committee Chair",
    "Industry_Sector": "Technology",
    "Company_Size": "Large-Cap",
    "Region": "North America",
    "High_Female_Representation": 1,
    "Balanced_Leadership": 1
}
```

11.2 cURL Example

You can test the endpoint from your terminal using curl:

```
curl -X POST http://127.0.0.1:5000/predict \
-H "Content-Type: application/json" \
-d '{
    "Age": 58, "Tenure_in_Years": 7, "Committee_Memberships": 1, "Education_Level": "Bachelor\"s Degree",
    "Total_Board_Members": 9, "Female_Board_Percentage": 11.11, "Avg_Board_Age": 55.2,
    "Avg_Board_Tenure": 4.1, "Leadership_Positions": 3, "PhD_Percentage": 22.22,
    "Gender_Diversity_Score": 22.22, "Is_Executive": 1, "Is_Leadership_Role": 1,
    "Board_Role": "CEO", "Industry_Sector": "Energy", "Company_Size": "Large-Cap",
    "Region": "Emerging Markets", "High_Female_Representation": 0, "Balanced_Leadership": 0
  }'
```

11.3 Output Format

A successful prediction request will return a JSON response like this:

```
{
  "confidence": {
    "female": 0.0551234567,
    "non_female": 0.9448765432
  },
  "input_summary": {
    "age": 58,
    "education": "Bachelor's Degree",
    "industry": "Energy",
    "role": "CEO",
    "tenure": 7
  },
  "prediction": "Non-Female",
  "prediction_numeric": 0,
  "timestamp": "2023-10-27T10:30:00.123456"
}
```

12 Conclusion

This application serves as a powerful demonstration of how machine learning can be applied to analyze complex social and corporate dynamics like gender diversity in leadership. The provided API is a robust tool for exploring the interplay of various factors—such as age, tenure, industry, and education—in predicting gender representation.

By following this manual, developers and data scientists can successfully set up, run, and interact with the prediction service. The project is a solid foundation that can be extended with real-world data, more advanced models, or a more feature-rich interactive dashboard.