**CHAPTER 4**

**DEVELOPMENT PROCESS**

* 1. **REQUIREMENT ANALYSIS**

Requirements are a feature of a system or description of something that the system is capable of doing in order to fulfil the system’s purpose. It provides the appropriate mechanism for understanding what the customer wants, analysing the needs assessing feasibility, negotiating a reasonable solution, specifying the solution unambiguously, validating the specification and managing the requirements as they are translated into an operational system.

* + 1. **PYTHON:**

Python is a dynamic, high level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural oriented programming. In Python, we don’t need to declare the type of variable because it is a dynamically typed language.

For example, x=10 .Here, x can be anything such as String, int, etc.

Python is an interpreted, object-oriented programming language similar to PERL, that has gained popularity because of its clear [syntax](https://whatis.techtarget.com/definition/syntax)and readability. Python is said to be relatively easy to learn and portable, meaning its statements can be interpreted in a number of [operating system](https://whatis.techtarget.com/definition/operating-system-OS)s, including UNIX-based systems, Mac OS, MS-DOS, OS/2, and various versions of Microsoft Windows 98. Python was created by Guido van Rossum, a former resident of the Netherlands, whose favourite comedy group at the time was Monty Python's Flying Circus. The source code is freely available and open for modification and reuse. Python has a significant number of users.

**Features in Python**

There are many features in Python, some of which are discussed below

* Easy to code
* Free and Open Source
* Object-Oriented Language
* GUI Programming Support
* High-Level Language
* Extensible feature
* Python is Portable language
* Python is Integrated language
* Interpreted Language
  1. **ANACONDA**

Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from [PyPI](https://en.wikipedia.org/wiki/Python_Package_Index" \o "Python Package Index) as well as the ana[conda](https://en.wikipedia.org/wiki/Conda_(package_manager)) package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command line interface (CLI).

The big difference between anaconda and the [pip package manager](https://en.wikipedia.org/wiki/Pip_(package_manager)) is in how package dependencies are managed, which is a significant challenge for Python data science and the reason anaconda exists.

When pip installs a package, it automatically installs any dependent Python packages without checking if these conflict with previously installed packages. It will install a package and any of its dependencies regardless of the state of the existing installation. Because of this, a user with a working installation of, for example, Google Tensorflow, can find that it stops working having used pip to install a different package that requires a different version of the dependent numpy library than the one used by Tensorflow. In some cases, the package may appear to work but produce different results in detail.

In contrast, anaconda analyses the current environment including everything currently installed, and, together with any version limitations specified (e.g. the user may wish to have Tensorflow version 2,0 or higher), works out how to install a compatible set of dependencies, and shows a warning if this cannot be done.

Opensource packages can be individually installed from the Anaconda repository, Anaconda Cloud (anaconda.org), or the user's own private repository or mirror, using the anaconda install command. Anaconda, Inc. compiles and builds the packages available in the Anaconda repository itself, and provides binaries for Windows 32/64 bit, Linux 64 bit and MacOS 64-bit. Anything available on [PyPI](https://en.wikipedia.org/wiki/Python_Package_Index" \o "Python Package Index) may be installed into a anaconda environment using pip, and anaconda will keep track of what it has installed itself and what pip has installed.

Custom packages can be made using the anaconda build command, and can be shared with others by uploading them to Anaconda Cloud, [PyPI](https://en.wikipedia.org/wiki/Python_Package_Index" \o "Python Package Index) or other repositories.

The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, it is possible to create new environments that include any version of Python packaged with anaconda.

* 1. **Anaconda Navigator**

Anaconda Navigator is a desktop [graphical user interface (GUI)](https://en.wikipedia.org/wiki/Graphical_user_interface) included in Anaconda distribution that allows users to launch applications and manage anaconda packages, environments and channels without using [command-line commands](https://en.wikipedia.org/wiki/Command-line_interface). Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for [Windows](https://en.wikipedia.org/wiki/Windows), [macOS](https://en.wikipedia.org/wiki/MacOS) and [Linux](https://en.wikipedia.org/wiki/Linux).

The following applications are available by default in Navigator:

* [JupyterLab](https://en.wikipedia.org/wiki/Project_Jupyter#JupyterLab)
* [Jupyter Notebook](https://en.wikipedia.org/wiki/Project_Jupyter#Jupyter_Notebook)
* QtConsole
* [Spyder](https://en.wikipedia.org/wiki/Spyder_(software))
* [Glue](https://en.wikipedia.org/wiki/Glue_(software))
* [Orange](https://en.wikipedia.org/wiki/Orange_(software))
* [RStudio](https://en.wikipedia.org/wiki/RStudio)
* [Visual Studio Code](https://en.wikipedia.org/wiki/Visual_Studio_Code)
  + 1. **JUPYTER NOTEBOOK**

Jupyter [Notebook](https://en.wikipedia.org/wiki/Notebook_interface) (formerly IPython Notebooks) is a [web-based interactive](https://en.wikipedia.org/wiki/Rich_Internet_application) computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter [web application](https://en.wikipedia.org/wiki/Web_application), Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a [JSON](https://en.wikipedia.org/wiki/JSON) document, following a versioned schema, containing an ordered list of input/output cells which can contain code, text (using [Markdown](https://en.wikipedia.org/wiki/Markdown)), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

Jupyter Notebook can connect to many kernels to allow programming in different languages. By default, Jupyter Notebook ships with the IPython kernel. As of the 2.3 release[[11]](https://en.wikipedia.org/wiki/Project_Jupyter#cite_note-releasenote23-11)[[12]](https://en.wikipedia.org/wiki/Project_Jupyter#cite_note-releasenote20-12) (October 2014), there are currently 49 Jupyter-compatible kernels for many programming languages, including [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), [R](https://en.wikipedia.org/wiki/R_(programming_language)), [Julia](https://en.wikipedia.org/wiki/Julia_(programming_language)) and [Haskell](https://en.wikipedia.org/wiki/Haskell_(programming_language)).

The Notebook interface was added to IPython in the 0.12 release[[14]](https://en.wikipedia.org/wiki/Project_Jupyter#cite_note-releasenote012-14) (December 2011), renamed to Jupyter notebook in 2015 (IPython 4.0 – Jupyter 1.0). Jupyter Notebook is similar to the notebook interface of other programs such as [Maple](https://en.wikipedia.org/wiki/Maple_(software)), [Mathematica](https://en.wikipedia.org/wiki/Mathematica), and [SageMath](https://en.wikipedia.org/wiki/SageMath" \o "SageMath), a computational interface style that originated with Mathematica in the 1980s. According to [The Atlantic](https://en.wikipedia.org/wiki/The_Atlantic), Jupyter interest overtook the popularity of the Mathematica notebook interface in early 2018.

* 1. **RESOURCE REQUIREMENTS:**

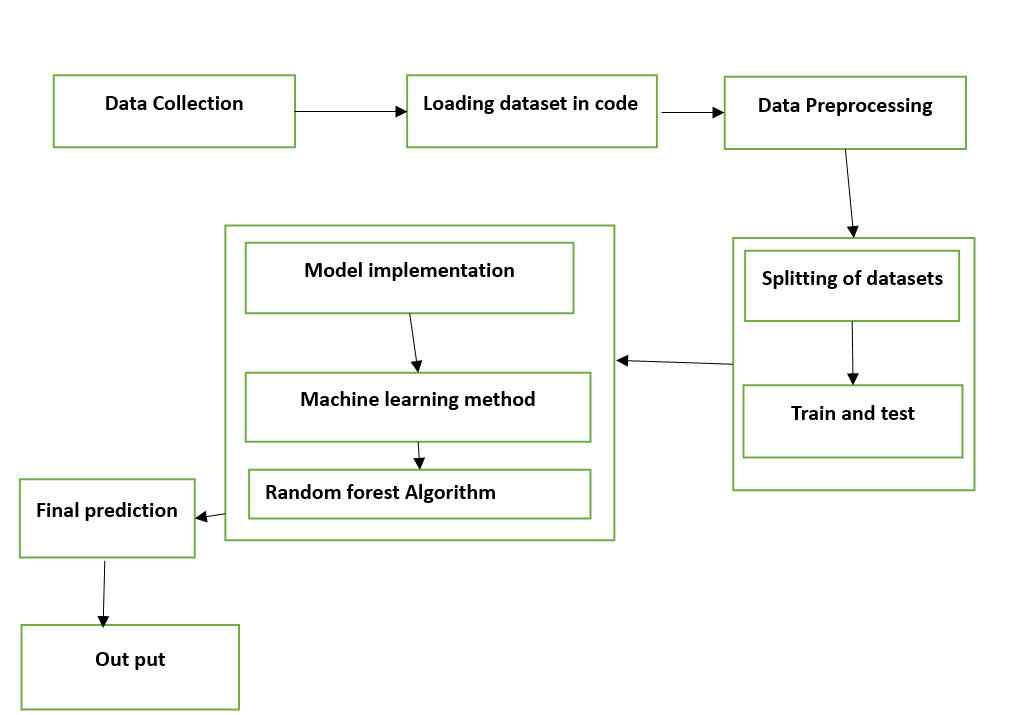
**SOFTWARE REQUIREMENTS**:

|  |  |
| --- | --- |
| Operating System | Windows 7or later |
| Simulation Tool | Visual Studio Code |
| Documentation | Ms – Office |

**HARDWARE REQUIREMENTS:**

|  |  |
| --- | --- |
| CPU type | I5 and Above |
| Ram size | 4GB |
| Hard disk capacity | 80 GB |
| Keyboard type | Internet keyboard |
| Monitor type | 15 Inch colour monitor |
| CD -drive type | 52xmax |

* 1. **SYSTEM ARCHITECTURE**

****

* 1. **PROPOSED SYSTEM**

Our proposed system utilizes a Random Forest algorithm, a powerful ensemble learning technique in machine learning, to analyze multiple facets of work culture. The system integrates various data sources such as employee surveys, HR records, performance metrics, and sentiment analysis from communication channels to construct a comprehensive understanding of the organizational environment. Through feature engineering and selection techniques, relevant variables influencing work culture are identified and fed into the Random Forest model.The Random Forest model then performs predictive analytics to assess the impact of different factors on overall work culture. By analyzing patterns within the data, the model can identify correlations, trends, and predictive indicators related to employee satisfaction, engagement, and productivity. Additionally, the model can highlight areas of concern such as communication breakdowns, leadership effectiveness, or work-life balance issues.The system provides interactive dashboards and visualization tools to present the insights derived from the Random Forest model in an intuitive and accessible manner. Decision-makers within the organization can explore key metrics, trends, and recommendations to guide strategic initiatives aimed at improving work culture. Furthermore, the system supports ongoing monitoring and evaluation, allowing companies to track progress over time and iterate on interventions effectively.

**ADVANTAGES:**

* Enhanced Predictive Accuracy By integrating deep learning with the Random Forest algorithm, the proposed hybrid approach leverages the strengths of both methodologies to achieve enhanced predictive accuracy. Deep learning models excel at automatically extracting complex features from raw data, enabling them to capture subtle patterns and relationships
* Robustness to Overfitting Random Forest is known for its robustness to overfitting, thanks to its ensemble learning strategy and the use of bootstrapping and feature randomness during training.
* Interpretability and Insights One of the key advantages of Random Forest is its ability to provide insights into feature importance, allowing clinicians and researchers to understand the factors contributing to the model's predictions.

**SYSTEM MODULES:**

Module 1 : Data collection

Module 2 : Data Pre processing

Module 3 : Model implementation

Module 4 : Loading the trained model

Model 5: prediction

**Module 1: Data Collection**

* In this module, you'll collect data related to work culture from various sources. This could include surveys, employee feedback, HR reports, performance reviews, and any other relevant data sources.
* The data should cover a range of features such as employee demographics, job satisfaction, work-life balance, company policies, leadership styles.

**Module 2: Data Preprocessing**

* Once the data is collected, it needs to be preprocessed before feeding it into the machine learning model.
* This involves tasks such as handling missing values, encoding categorical variables, scaling numerical features, and possibly feature engineering to create new informative features.

**Module 3: Model Implementation**

* In this module, you'll implement the random forest algorithm for predicting work culture based on the preprocessed data.
* Random forest is a popular ensemble learning technique that builds multiple decision trees and combines their predictions to improve accuracy and robustness. You'll use libraries like scikit-learn in Python to build and train the random forest model.

**Module 4: Loading the Trained Model**

* After training the random forest model, you'll save it to disk for future use.
* This module involves writing code to serialize the trained model into a file format that can be easily loaded later.
* This step is crucial for deploying the model in production environments where it can make predictions on new data.

**Module 5: predictions**

* you'll utilize the trained random forest model to make predictions on new data instances.
* This could involve loading the saved model, preprocessing new data in the same way as the training data, and then using the model to predict the work culture of a company based on the input features.
* The output of this module could be insights or recommendations for improving work culture based on the predictions made by the model.