# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

**"JNANA SANGAMA" BELAGAVI - 590 018**

**KARNATAKA**

## REPORT OF INDUSTRY

Carried out in

**SEVENTH SENSE TALENT SOLUTIONS , BANGALORE**

**SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF**

**BACHELOR OF ENGINEERING**

**IN**

**ARTIFICIAL INTELLIGENCE & DATA SCIENCE**


Submitted by:

**MD TANZIL MASUD**

**[1CG21AD023]**

**INTERNAL GUIDE**

**Mr. Dharneshkumar M L,M.Tech**
**Assistant Profess**
**Dept. of AD,**
**C.I.T, Gubbi, Tumkur.**

**EXTERNAL GUIDE**

**Mr. Dikshith N M**
**Project Manager**
**Seventh Sense Talent Solution**
**Bangalore**

**HOD**

**Dr. Gavisiddappa Ph.D**
**Professor & Head,**
**Dept. of AD**
**CIT, Gubbi**

**Channabasaveshwara Institute of Technology**
**(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)**
**(NAAC Accredited & ISO 9001:2015 Certified Institution)**
NH 206 (B.H. Road), Gubbi, Tumkur – 572216. Karnataka


**DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE**

**2024-2025**

**Channabasaveshwara Institute of Technology**
**(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)**
**(NAAC Accredited & ISO 9001:2015 Certified Institution)**
NH 206 (B.H. Road), Gubbi, Tumkur – 572216. Karnataka.

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE**
**2024-2025**

# CERTIFICATE

This is to certify that the internship entitled "**AUTOMATED CODE GENERATION FROM NATURAL LANGUAGE DESCRIPTIONS**" has been carried out by **MD TANZIL MASUD - [1CG21AD023]** bonafide student of **CHANNABASAVESHWARA INSTITUTE OF TECHNOLOGY, GUBBI, TUMKUR**, in partial fulfillment of the requirement for the award of the degree **Bachelor of Engineering** in **ARTIFICIAL INTELLIGENCE & DATA SCIENCE** from the **Visvesvaraya Technological University, Belagavi** during the year **2024-2025.** It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report. The Internship report has been approved as it satisfies the academic requirements in respect of Internship prescribed for the said degree.

Signature of Guide

**Mr.Dharneshkumar M L,**M.Tech
Assistant Professor,
Dept., of AD
C.I.T, Gubbi.

Signature of HOD

**Dr. Gavisiddappa** Ph.D
Professor & Head,
Dept., of AD
C.I.T, Gubbi.

Signature of Principal

**Dr. SURESH D S** Ph.D
Director & Principal
C.I.T, Gubbi.

External Viva

| Examiners Name | Signature with Date |
|---|---|
| 1. _____ | _____ |
| 2. _____ | _____ |

**Channabasaveshwara Institute of Technology**
**(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)**
**(NAAC Accredited & ISO 9001:2015 Certified Institution)**
NH 206 (B.H. Road), Gubbi, Tumkur – 572216. Karnataka.

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE**
**2024-2025**

# UNDERTAKING

I, **MD TANZIL MASUD** bearing **1CG21AD023**, student of **VII Semester B.E.** in **ARTIFICIAL INTELLIGENCE & DATA SCIENCE, C.I.T, GUBBI, TUMKUR** hereby declare that the Internship carried out in **SEVENTH SENSE TALENT SOLUTIONS , BANGALORE** and submitted in partial fulfillment of the requirements for the award of the degree **Bachelor of Engineering** in **ARTIFICIAL INTELLIGENCE & DATA SCIENCE** of the **Visvesvaraya Technological University, Belagavi** during the academic year 2024-2025.

**Place: GUBBI**
**Date:**

**MD TANZIL MASUD**
**1CG21AD023**

**Channabasaveshwara Institute of Technology**
**(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)**
**(NAAC Accredited & ISO 9001:2015 Certified Institution)**
NH 206 (B.H. Road), Gubbi, Tumkur – 572216. Karnataka.

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE**

**2024-25**

# BONAFIDE CERTIFICATE

This is to certify that the Internship carried out in **SEVENTH SENSE TALENT SOLUTIONS , BANGALORE** is a bonafide work of **MD TANZIL MASUD – [1CG21AD023],** student of **VII** semester **B.E.- ARTIFICIAL INTELLIGENCE & DATA SCIENCE** from **Channabasaveshwara Institute of Technology, Gubbi, Tumkur,** in partial fulfillment of the requirements for the award of degree **B.E.,** in **ARTIFICIAL INTELLIGENCE & DATA SCIENCE** of **Visvesvaraya Technological University**, **Belagavi** during the academic year 2024-2025. It is certified that the Internship work carried out was under my supervision and guidance.

Guide

Mr.Dharneshkumar M L,M.Tech
Assistant. Professor
Dept., of AD
C.I.T, Gubbi.

# ACKNOWLEDGEMENT

Several special people have contributed significantly to this effort. First of all, I am grateful to my institution, **Channabasaveshwara Institute of Technology, Gubbi**, which provides me an opportunity in fulfilling my most cherished desire of reaching my goal.

I, acknowledge and express my sincere thanks to our beloved Director & Principal, **Dr. Suresh D S**, for his many valuable suggestion and continued encouragement by supporting me in mt academic endeavors.

I, express my sincere gratitude to **Dr. Gavisiddappa**, Professor and Head, Department of AD, for providing his constructive criticisms and suggestions.

I, extend my gratitude to my Internship guide **Mr.Dharneshkumar ML,M.Tech, Assistant Professor,** Department of AD, for his guidance, support and suggestions throughout the period of this Internship.

I express my deep sense of gratitude to **SEVENTH SENSE TALENT SOLUTIONS , BANGALORE** for giving such an opportunity to carry out the internship in their esteemed industry.

I sincerely thank **Mr. Dikshith NM, Project Manager , SEVENTH SENSE TALENT SOLUTION,BANGALORE** for exemplary guidance and supervision.

Finally, I would like to thank all the individuals who supported me directly and indirectly for the successful completion of this internship work.

**MD TANZIL MASUD[1CG21AD023]**

# ABSTRACT

Automated Code Generation Using NLP Descriptions leverages advanced Natural Language Processing (NLP) techniques and transformer-based machine learning models to bridge the gap between natural language and programming.

This project focuses on translating plain English descriptions into functional Python code using a pre-trained model, Salesforce CodeGen, integrated with a user-friendly Flask-based API. By inputting a natural language description, users can obtain executable code snippets in real-time, streamlining the coding process and making programming more accessible.The system preprocesses descriptions through tokenization and encodes them into a format compatible with the CodeGen model, which generates code tokens based on learned patterns. The output is then decoded into human-readable Python code. .This project highlights the potential of AI-driven tools to revolutionize software development, empowering users to automate repetitive coding tasks and focus on creative problem-solving.

# TABLE OF CONTENT

# CHAPTER 1:

## COMPANY PROFILE

Seventh sense talent solutions offers consulting, recruitment, and learning/training solutions to top companies in India. The company was founded by IIM Bangalore alumni and helps clients with talent acquisition, development, and management.

The vision is to help students secure their dream jobs and careers and that is the guarantee we provide through our services. We are a team that believes that cost effective and high quality content coupled with intelligent test packages can help any student succeed

Seventh Sense's intelligent and adaptive portal provides everything for success in exams - detailed concepts in each area of the examination, sectional quizzes, comprehensive tests, easy to understand solutions leading to continual improvement in every student - Helping student improve on their strengths and develop their weak areas.

Beyond conventional training programs, the organization distinguishes itself through impactful projects, collaborating with government bodies and industry leaders. Noteworthy initiatives include being the Placement Partner for the Government of Karnataka, Lead Implementation Partner for 'Empowerment for Her' and 'Skills for Her' projects sponsored by Birlasoft, Accenture, and NASSCOM Foundation, among others.

The company's involvement extends to diverse regions, encompassing Karnataka, Delhi NCR, UP, Maharashtra, Tamil Nadu, Telangana, Kerala, Gujarat, and more. Seventh Sense is a key participant in projects fostering youth-led entrepreneurship, digital literacy, and skill-building across multiple states, collaborating with UNDP, NSDC, CDOT, ITC, and UNICEF. Through its multifaceted approach, Seventh Sense People Development Solutions remains dedicated to shaping a skilled and empowered workforce across the nation.

We count among our clients some of the top 100 colleges and top 50 companies in India and have served clients across the country. We have served students from colleges such as IIM Bangalore, IISc, RVCE, PES, RNSIT, UVCE, SIT, SDM, CMR, Reva, RVR&JC, Vignan, SVCE, MVSR, GPREC, BNMIT (indicative). We have also served corporates such as Hindustan UniLever, Tata Technologies, General Electric, SAP India Consulting, Deloitte & Touche, ABB, Gammon, OnMobile and GMR Ltd.

**Services**

- Women Empowerment
- Skilling
- Employement
- Research
- Innovation & Entrepreneurship

# CHAPTER 2:

## INTRODUCTION

Programming has become an essential skill in today's technology-driven world. However, for non-technical individuals, translating their ideas into executable code remains a significant challenge. Even for experienced developers, repetitive tasks such as writing boilerplate code or implementing utility functions can consume valuable time and hinder productivity. Automated code generation bridges this gap by utilizing Artificial Intelligence (AI) and Natural Language Processing (NLP) to translate natural language descriptions into functional programming code. This project aims to leverage advancements in transformer-based models like Salesforce CodeGen to build a system capable of generating accurate Python code snippets from plain English descriptions. Despite the rise of tools like code completion plugins in Integrated Development Environments (IDEs), these solutions primarily assist developers by offering suggestions for already written code rather than generating new code from scratch. The current need is for a system that allows users to input requirements in natural language and automatically receive relevant, functional code. This approach reduces barriers for non-technical users and improves efficiency for developers by automating routine coding tasks.

The primary objective of this project is to create a prototype system that can understand natural language inputs and generate syntactically correct and contextually relevant Python code. It also aims to integrate this functionality into a user-friendly RESTful API built with Flask, enabling real-time interaction with the model. The system uses `Salesforce/codegen-350M-mono`, a pre-trained transformer-based model, fine-tuned to perform text-to-code tasks. The generated code snippets can be evaluated using metrics such as BLEU scores, which assess the similarity between the generated code and expected outputs. The scope of this project is limited to Python as the target programming language due to its simplicity, versatility, and widespread use across various domains such as data science, automation, and web development. The project focuses on generating functions or small code snippets based on structured and specific natural language inputs. While the system has potential applications in other programming languages, the current implementation will remain Python-centric.

This project has several real-world applications. For beginners, the system serves as an educational tool, helping them learn programming by providing examples of functional code

based on plain English descriptions. For experienced developers, it accelerates software development by automating the generation of repetitive code. Business professionals or data analysts with limited programming knowledge can also benefit from the system, as it allows them to implement basic functionality without writing code manually. The challenges of implementing such a system are multifaceted. Natural language descriptions can often be ambiguous or incomplete, leading to difficulties in generating accurate code. Moreover, pre-trained models like Salesforce CodeGen, while powerful, are not infallible. They may produce incorrect, incomplete, or even insecure code in certain scenarios. Additionally, large models require significant computational resources, posing scalability challenges. Evaluating the quality of generated code is another hurdle, as traditional metrics like BLEU scores might not effectively capture functional correctness.

# CHAPTER 3:

# INTERNSHIP TASK

## 3.1 Tools & Technologies

- Streamlit: For building and deploying an interactive web application to generate machine learning starter code.
- Jinja2: For templating and dynamically generating Python code.

### 3.1.1 Programming Language

- Python: The core language for implementing the functionality and ML code generation.

### 3.1.2 Frameworks and Libraries

I. Frontend and User Interface

- Streamlit: Used to create an intuitive and interactive GUI for selecting ML tasks, algorithms, and configurations.

II. Backend and Utility

- Jinja2: For rendering templates and generating code dynamically.

III. Sidebars

- Custom sidebar scripts (e.g., sidebar.anomaly_detection_sidebars, sidebars.classification_sidebars,etc) to configure algorithm-specific inputs.

IV. Miscellaneous

- Base64: For encoding the generated code to enable download functionality.
- os: For handling file paths and interactions with the file system.

### 3.1.3 Data Handling

- User Input: Using Streamlit sidebars for collecting task, algorithm, and hyperparameter inputs.
- Template Files: Storing pre-defined ML code templates in structured directories (e.g., templates/Anomaly Detection/LOF).

- Dynamic Rendering: Generating Python code from Jinja2 templates

### 3.1.4 Development Tools

- Text Editor/IDE: (e.g., Visual Studio Code, PyCharm) for writing and testing Python code.
- Version Control: (e.g., GitHub) for managing and sharing the codebase, as inferred from the badges.
- Streamlit CLI: For running the application locally (streamlit run <filename>).

### 3.1.5 Deployment Tools

- Streamlit Cloud: For deploying the application and making it accessible via a web interface.
- GitHub: For hosting the project repository.

## 3.2 Task Title :Automated Code Generation Using NLP Descriptions

### 3.2.1 Objectives

- To provide users with a user-friendly interface for generating machine learning starter code.
- To reduce the time and effort required to set up machine learning models.
- To support anomaly detection, classification, and clustering tasks.
- To offer customization of datasets and hyperparameters.
- To allow users to download ready-to-use Python scripts.

### 3.2.2 System Design

The system is built using Streamlit to provide an interactive web interface. The template-based code generation is handled using Jinja2, and modular sidebars are used for user inputs.

Key Components

- Frontend: Built using Streamlit for user interactions.
- Backend: Uses Jinja2 templates to dynamically generate code.

- Sidebars: Organized into modular input forms for different machine learning tasks and algorithms.

### 3.2.3 Input Module

- Task Selection: Users choose a task (e.g., Anomaly Detection, Classification, or Clustering).
- Algorithm Selection: Users select an algorithm corresponding to the chosen task.
- Hyperparameter Input: Dynamic input fields for hyperparameter configuration based on the selected algorithm.
- Dataset Input: Users specify dataset paths or upload datasets for training.

### 3.2.4 Processing Module

I. Template Mapping:

- Maps the selected task and algorithm to a predefined Jinja2 template.
- Templates are stored in the templates/ directory, organized by task and algorithm.

II. Template Rendering:

- Uses the Jinja2 library to render Python code by passing user inputs and configurations.

III. Error Handling:

- Validates inputs and provides feedback in case of missing or incorrect values.

### 3.2.5 Output Module

I. Generated Code: Displays the generated Python code in the main panel.

II. Download Option:

- Provides a downloadable link for the Python file.
- File naming format: <task>_<algorithm>.py (e.g., classification_svm.py).

III. Code Preview:

- Uses st.code() to preview the generated script in the app.

### 3.2.6 System flow

- User Interaction: Users interact with the frontend to choose tasks, algorithms, and input parameters.
- Sidebar Inputs: Modular sidebars collect user inputs specific to the selected task and algorithm.
- Template Mapping: The backend maps user selections to appropriate Jinja2 templates.
- Code Generation: The backend dynamically generates Python scripts using the templates and inputs.
- Code Preview & Download:The final script is displayed for review and made available for download.

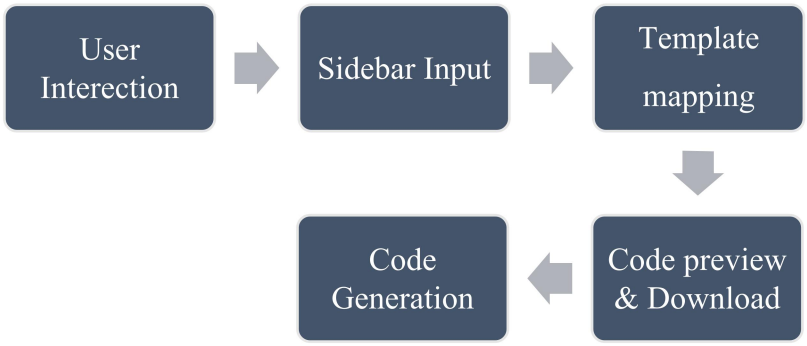


**Fig 3.2.6 System Flow Diagram**

### 3.2.7 Data Flow

- User: Initiates interaction by providing inputs such as selecting tasks, algorithms, and dataset details.
- Streamlit UI: Captures user inputs via the interface and sidebars.
- Input Module: Processes user inputs, ensuring they are correctly formatted and complete.
- Processing Module: Maps inputs to appropriate templates, renders the Python code, and handles errors if necessary.
- Output Module: Displays the generated Python code, provides a preview, and offers a downloadable link for the script.

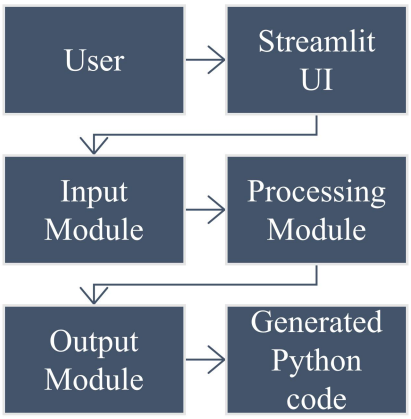


**Fig 3.2.7 Data Flow Model**

### 3.2.8 Sequence Diagram

- User selects task: The user chooses the task they want to perform (e.g., training a model, predicting outcomes).
- User selects algorithm: The user picks the algorithm that suits the task (e.g., K-Nearest Neighbors, Decision Trees).
- User enters dataset and hyperparameters: The user provides the dataset and necessary hyperparameters (e.g., learning rate, number of trees).
- System renders template: The system generates a code template based on the user's inputs.
- Template returns generated code: The system returns the generated code to implement the task.
- System displays and allows code download: The system shows the code to the user and offers the option to download it.

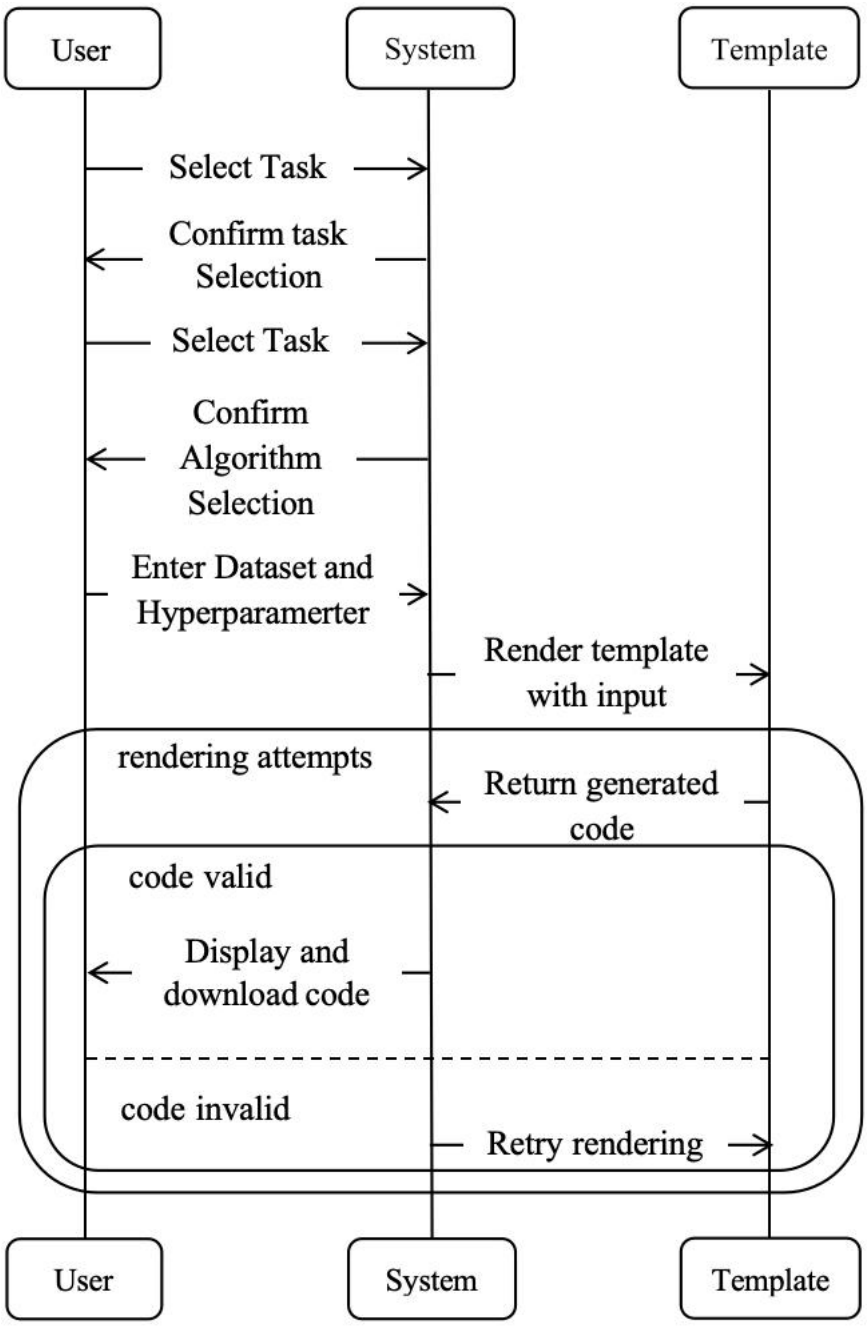


**Fig 3.2.8 Sequence Diagram**

### 3.2.9 Implementation Details

I. Data Preprocessing

- The system does not preprocess datasets directly.
- Users can specify paths to preprocessed datasets.

II. Implementation Modules

- Streamlit Interface: Handles user interactions and layout.

- Jinja2 Templates:
  - Dynamic templates for each task and algorithm.
  - Templates use placeholders for user inputs.
- Sidebar Modules: Modular forms for input collection based on tasks and algorithms.

III. Key Files

- main.py: Main Streamlit app file.
- sidebars/: Directory containing sidebar modules.
- templates/: Directory containing Jinja2 templates.

### 3.3.0 Algorithm

### I. Logistic Regression

Logistic regression is a supervised learning algorithm used for classification problems, particularly binary classification. Despite its name, it is a classification algorithm and not a regression algorithm.

- Basic Idea

The goal of logistic regression is to model the probability that a given input belongs to a particular class using a logistic function (sigmoid function). For binary classification, the model predicts probabilities between 0 and 1, which can then be thresholded to classify the input.

- Mathematical Formulation

(a) Hypothesis Function

Logistic regression predicts the probability P(y=1/X)P(y=1/X) using the sigmoid function:

$$h\theta(X)=\sigma(\theta^TX)=1/1+e^{-\theta^T X}$$

Where:

X: Input features (vector).

θ: Model parameters (weights).

σ(z): Sigmoid function.

(b) Sigmoid Function

The sigmoid function maps any real value to a range between 0 and 1:

$$\sigma(z)=1/1+e^{-z}$$

- Optimization

The objective is to minimize the cost function J(θ)J(θ). This is typically done using:

Gradient Descent:

- Update rule:

$$\theta=\theta-\alpha.\nabla J(\theta)$$

- α:Learning rate

Stochastic Gradient Descent (SGD) or Mini-batch Gradient Descent:

- For efficiency in large datasets.

Advanced Optimizers:

- Methods like Adam, RMSProp, or LBFGS are sometimes used.

- Binary Classification Decision Rule

The predicted probability hθ(X)hθ (X) is thresholded to make a class prediction:

- If hθ(X)≥0.5hθ,predict y=1y=1.
- Otherwise, predict y=0y=0.

The threshold can be adjusted based on the use case (e.g., lowering it for imbalanced datasets).

- Multiclass Classification

For multiclass problems, Logistic Regression is extended using:

One-vs-Rest (OvR):

- Train one logistic regression model per class.

Softmax Regression:

- Generalization of logistic regression using the softmax function.

- Applications
  - Medical diagnosis (e.g., predicting disease presence).
  - Spam detection in emails.
  - Customer churn prediction.
  - Sentiment analysis (positive/negative sentiment).
  - Credit scoring and fraud detection.

**II. LOF(Local Outlier Factor)**

The Local Outlier Factor (LOF) is an unsupervised anomaly detection algorithm that identifies anomalous data points by comparing their local density to the densities of their neighbors. It is especially useful for detecting outliers in datasets with complex structures.

- Basic Idea

LOF measures the degree of abnormality of a data point based on:

- How isolated it is compared to its neighbors.
- The density of the data points surrounding it.

Points with much lower local density compared to their neighbors are considered outliers.

- Key Concepts in LOF

(a) Distance Metrics

LOF relies on a distance metric (e.g., Euclidean distance) to calculate the proximity of points.

(b) Neighborhood Definition

The neighborhood of a point is defined by the kk-nearest neighbors (k-NN). The number kk is a user-defined parameter.

(c) Reachability Distance

The reachability distance between two points pp and oo is defined as:

$$reach\text{-}dist_k(p,o)=max\{k\text{-}distance(o),dist(p,o)\}$$

k-distance(o): Distance to the k-th nearest neighbor of o.

dist(p,o): Distance between p and o.

- Algorithm Steps

Choose Parameters:

- Number of neighbors kk.
- Distance metric (e.g., Euclidean, Manhattan).

Compute Distances:

- Calculate the kk-distance for each point.
- Determine the kk-nearest neighbors for each point.

Calculate Reachability Distances:

- For each point, compute the reachability distances to its neighbors.

Calculate Local Reachability Density (LRD):

- Compute the LRD for each point using its neighbors' reachability distances.

Compute Local Outlier Factor (LOF):

- Compare the LRD of each point to the LRDs of its neighbors to calculate the LOF.

Interpret Results:

- Points with a high LOF score (typically >1.5>1.5) are considered outliers.

- Applications

- Fraud detection in financial transactions.
- Intrusion detection in network security.
- Outlier detection in environmental data.
- Medical diagnosis (e.g., detecting rare diseases).
- Quality control in manufacturing.

### III. K-means

K-Means Clustering is an unsupervised learning algorithm used for clustering problems. It partitions the data into kk distinct non-overlapping subsets (clusters) based on feature similarity, minimizing intra-cluster variance while maximizing inter-cluster differences.

- Basic Idea

The goal of K-Means is to divide nn data points into kk clusters such that each data point belongs to the cluster with the nearest mean (centroid).

- Mathematical Formulation

(a) Objective Function

K-Means aims to minimize the sum of squared distances between each point and its corresponding cluster centroid:

$$J=\sum i=1k\sum x \in Ci \ ||x-\mu i||$$

Where:

k: Number of clusters.

Ci: Set of points in the ii-th cluster.

μi: Centroid of the ii-th cluster.

||x−μi^||2: Squared Euclidean distance between a point xx and its cluster centroid.

- Algorithm Steps

Initialize Centroids:

- Randomly select kk data points as the initial centroids.

Assign Clusters:

- For each data point, calculate the distance to all centroids and assign the point to the nearest centroid.

Update Centroids:

- For each cluster, compute the new centroid as the mean of all points assigned to that cluster:

$$\mu i=1/|Ci|\sum x \in Ci$$

Repeat:

Iterate between assigning clusters and updating centroids until:

- Centroids do not change significantly, or
- A predefined number of iterations is reached.

➢ Key Parameters

k: Number of Clusters

- Predefined by the user.
- Use methods like the Elbow Method or Silhouette Analysis to determine the optimal value.

Distance Metric

- Typically, Euclidean distance is used, but other metrics (e.g., Manhattan, Cosine) can be applied.

Initialization

- Random initialization can lead to suboptimal results.
- K-Means++ is a popular initialization method to improve performance.

➢ Applications

- Market segmentation (e.g., customer groups in e-commerce).
- Document clustering in text mining.
- Image compression and segmentation.
- Anomaly detection.
- Recommendation systems.

# CHAPTER 4:

# RESULTS

## 4.1

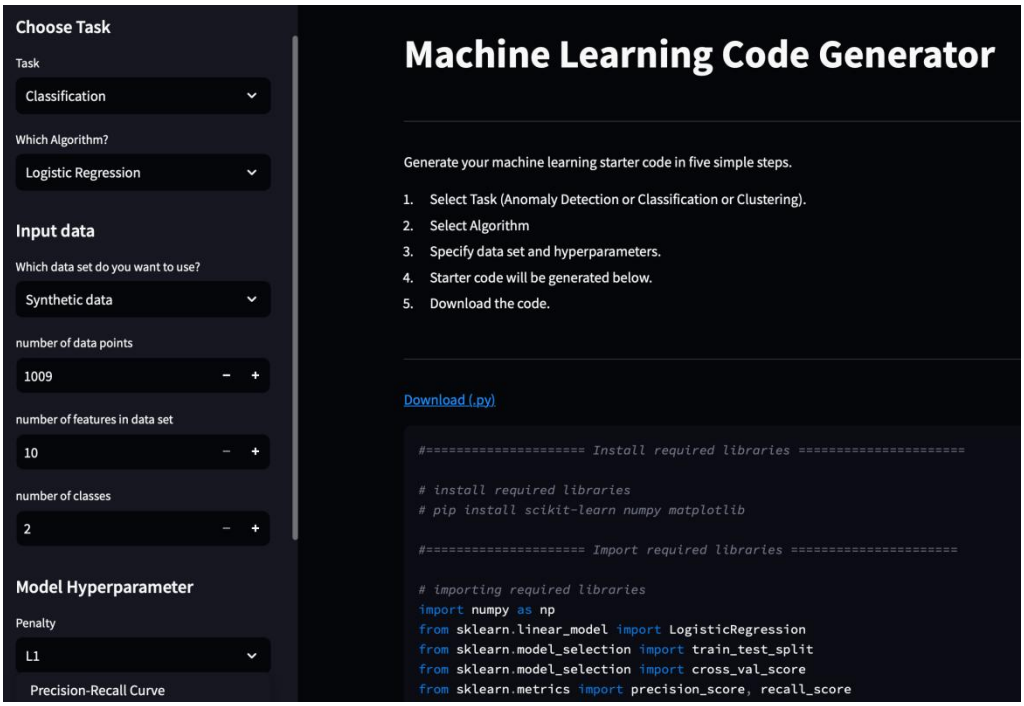


**Fig 4.1 Task selected and libraries install and imported**

## 4.2

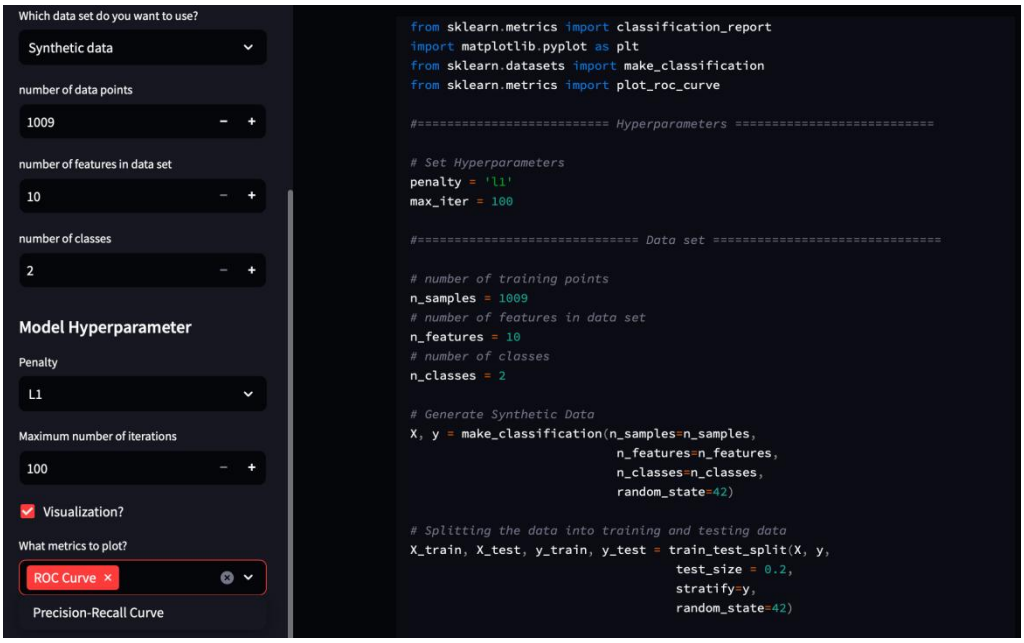


**Fig 4.2 Hyperparameter and Dataset is initialized**

## 4.3

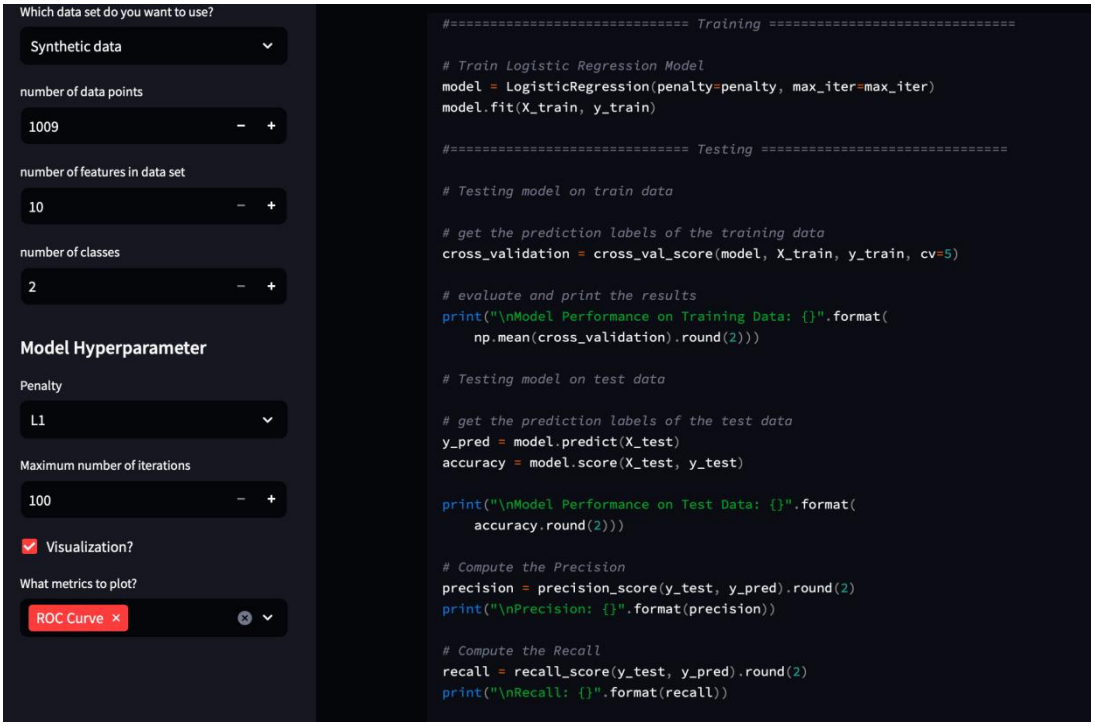


**Fig 4.3 Training and Testing is done**

## 4.4

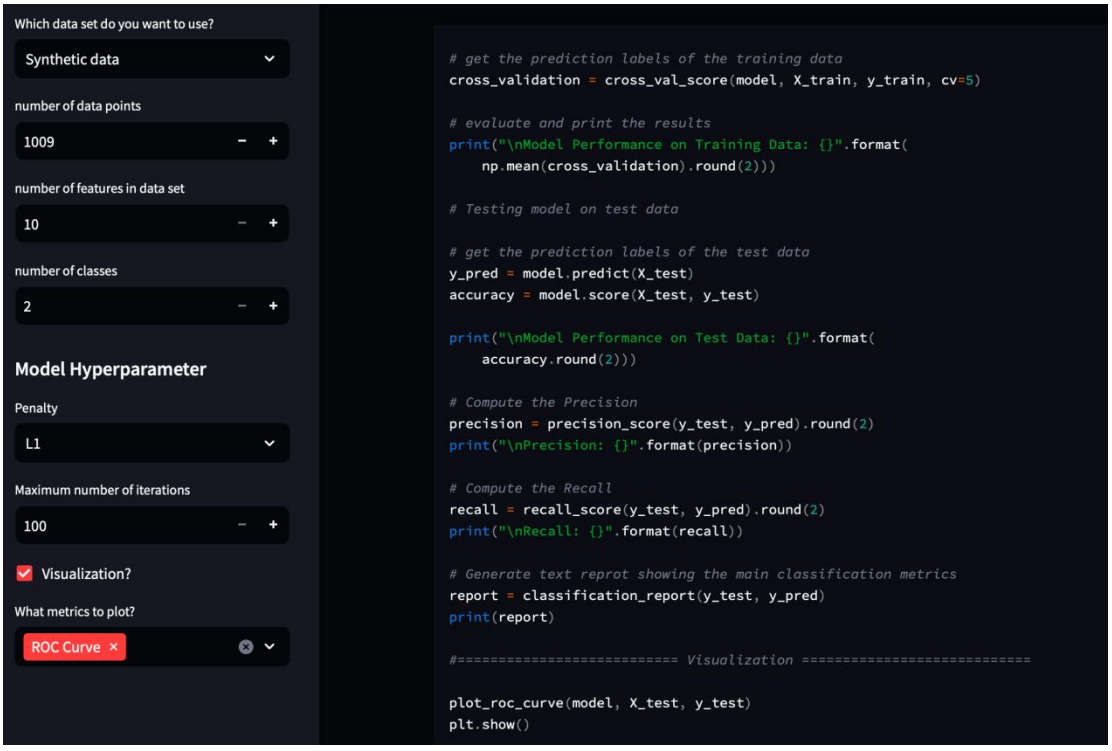


**Fig 4.4 Visualization**

## 4.5

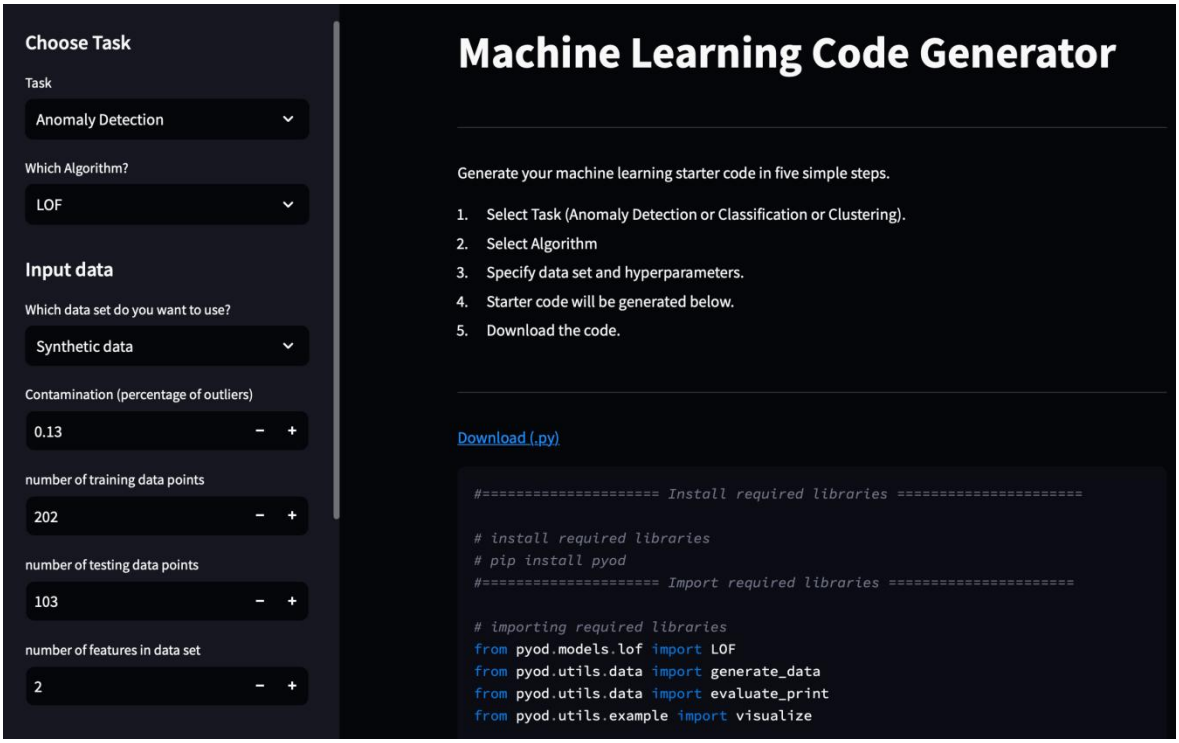


**Fig 4.5 Task selected and libraries install and imported**

## 4.6

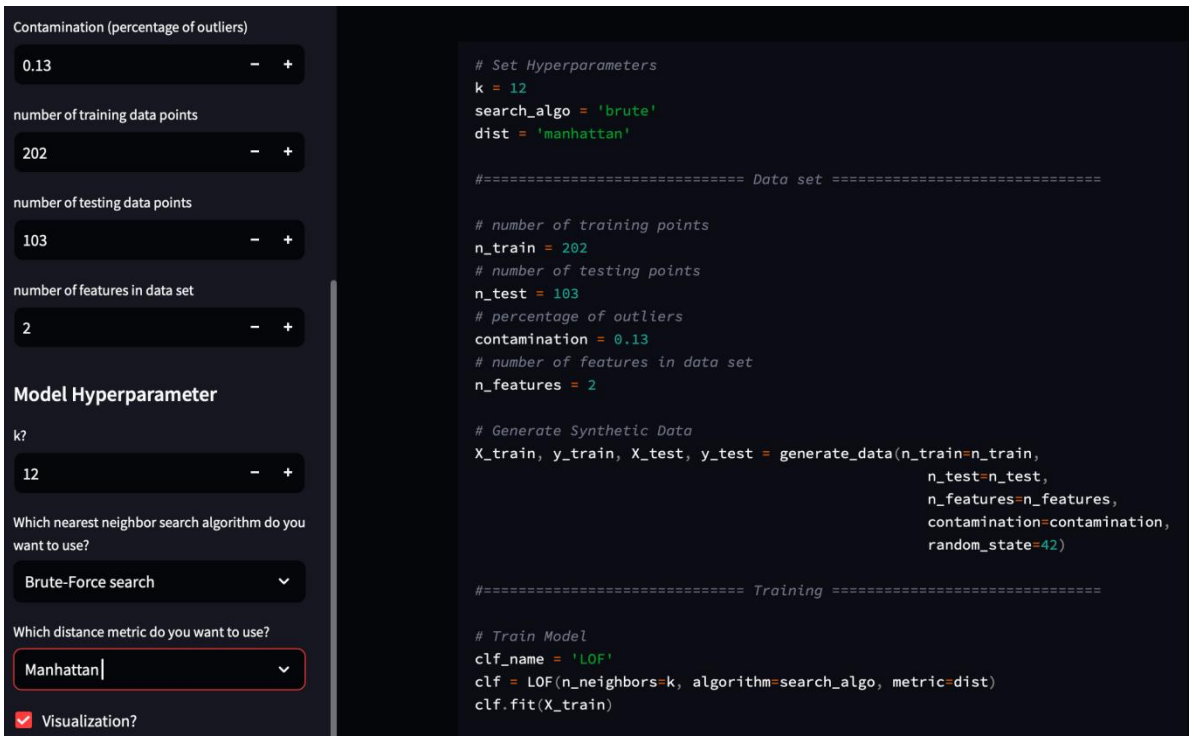


**Fig 4.6 Hyperparameter and dataset is set and model is also trained**

## 4.7

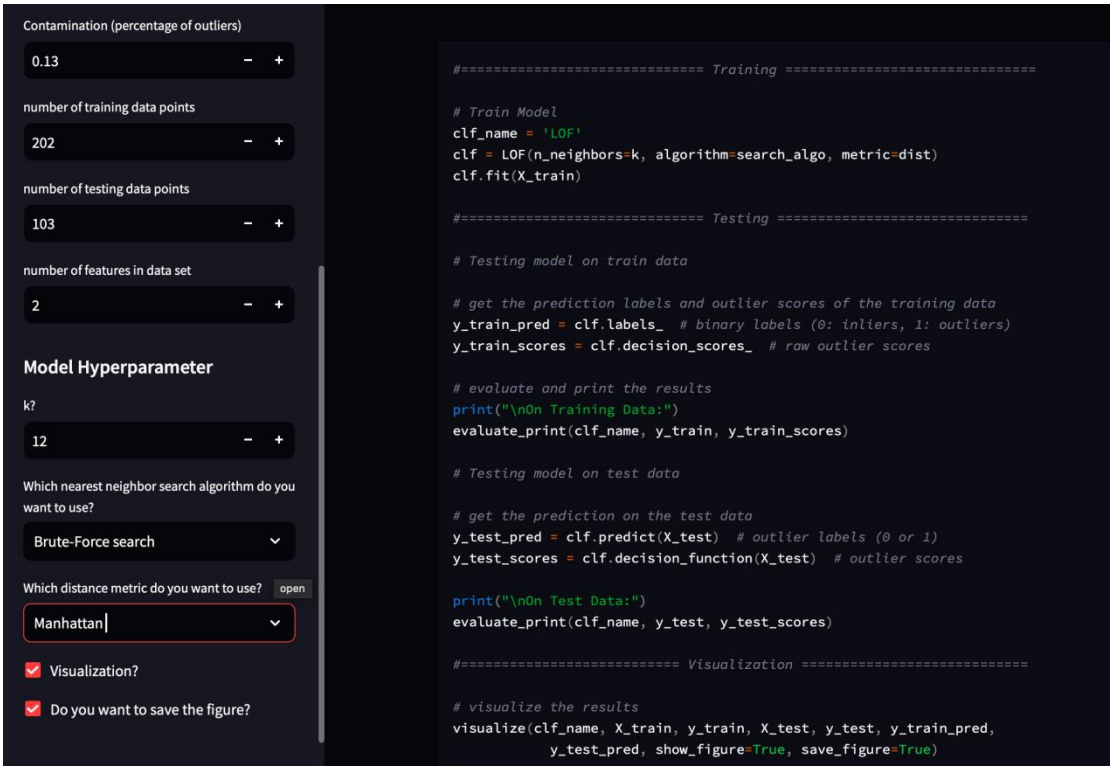


**Fig 4.7 Testing and Visualization**

## 4.8

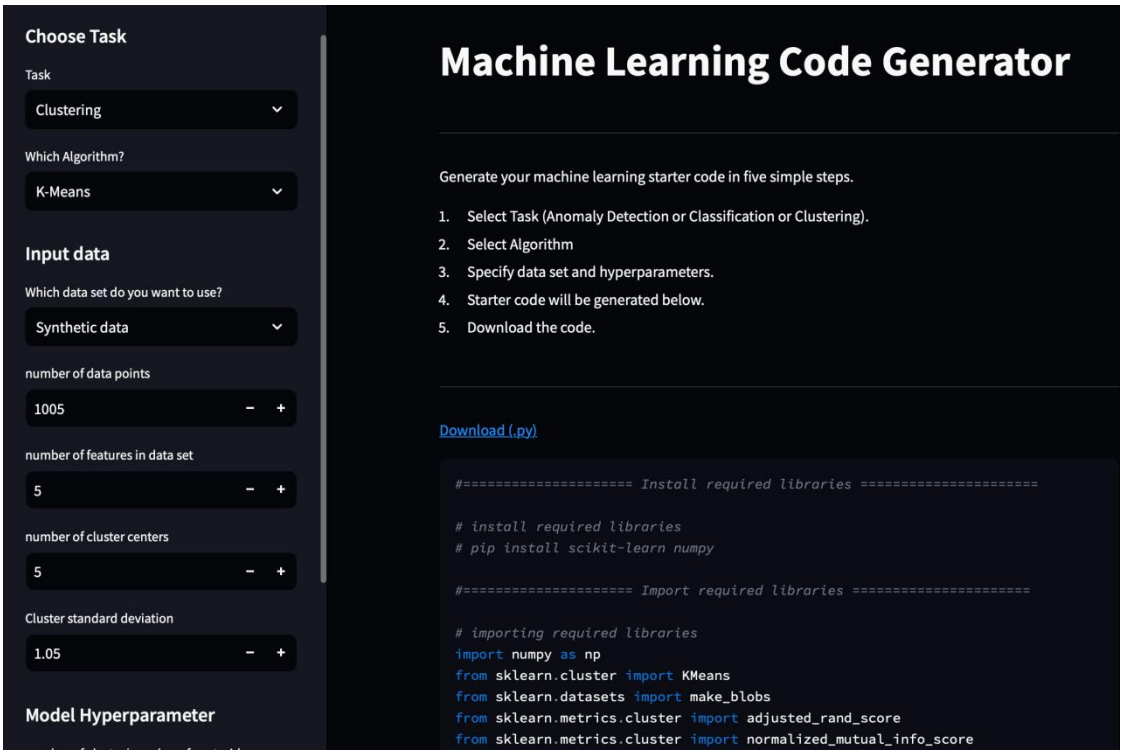


**Fig 4.8 Task selected and libraries install and imported**

## 4.9

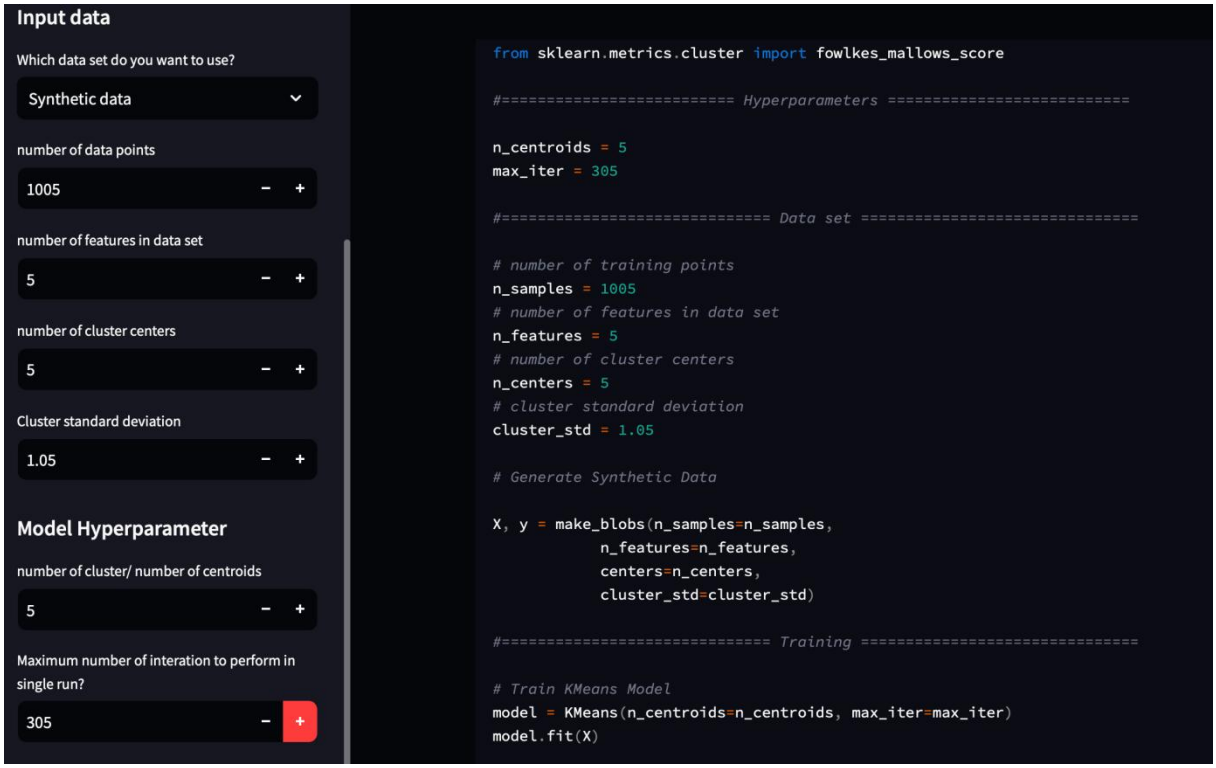


**Fig 4.9 Hyperparameter and dataset is set and model is also trained**

## 4.10

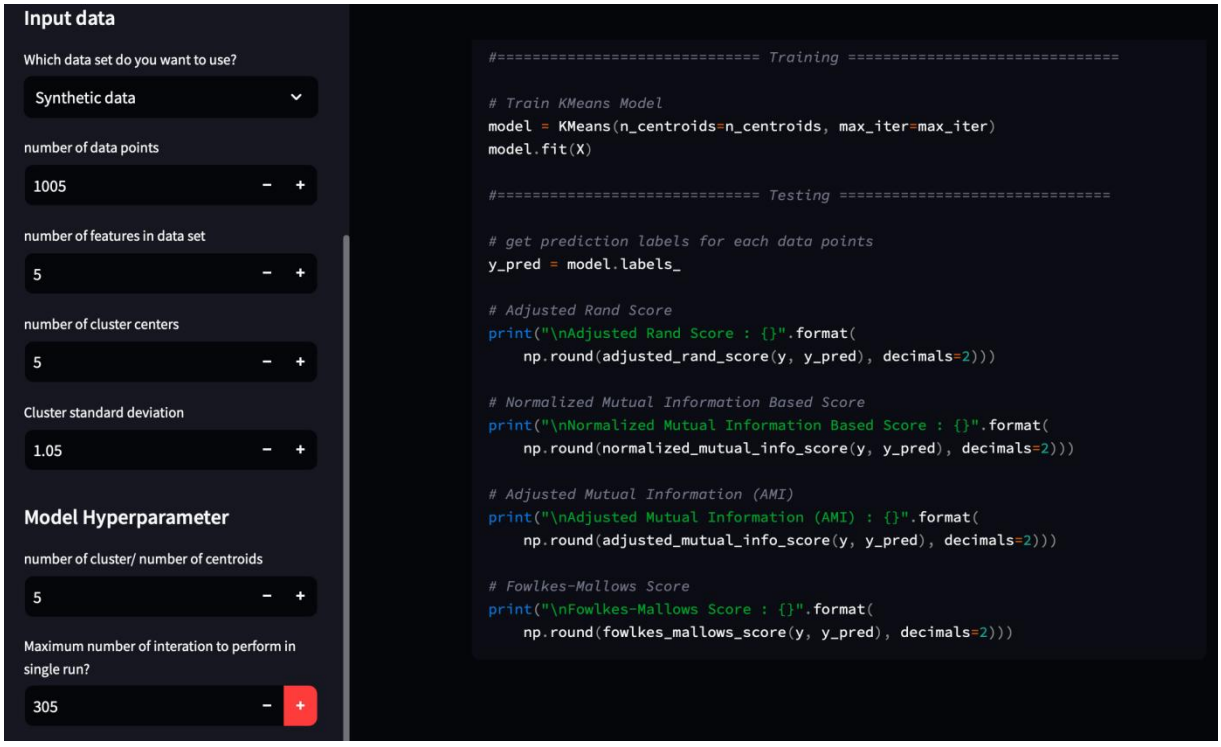


**Fig Training and Testing**

# CHAPTER 5:

## CONCLUSION

The Automated Code Generation Using NLP Descriptions project successfully demonstrated the integration of Natural Language Processing (NLP) and machine learning models to automate the process of generating Python code from plain English descriptions. By utilizing the Salesforce CodeGen model, a transformer-based architecture pre-trained on large-scale programming data, the system can effectively translate well-defined descriptions into syntactically correct and functional code. This approach holds great promise for reducing the time and effort required to write code manually, making coding more accessible to non-technical users while accelerating development for technical professionals.

Throughout the project, the system showed impressive performance, especially in handling simple tasks like arithmetic operations, sorting algorithms, and string manipulations. The Flask-based API provides a seamless interface for users to interact with the model, receiving generated code in real-time. The use of metrics like BLEU score and human evaluation demonstrated that the system was able to produce accurate code in most cases, with a high level of user satisfaction.

However, the project also faced several challenges. Handling ambiguous or vague descriptions proved to be difficult, as the model sometimes generated incorrect or incomplete code. Additionally, more complex, domain-specific tasks were not always well-understood by the model. While the system generated accurate code for common tasks, improvements in fine-tuning, dataset diversity, and model handling of complex scenarios are required to further enhance its capabilities.

Looking ahead, there are numerous opportunities for improvement and expansion. Future iterations could focus on increasing the model's ability to understand a wider variety of programming languages and handle more complex descriptions. Further advancements might also include integrating automated error checking, code optimization, and ensuring better scalability. By expanding the scope of tasks the system can handle and refining its capabilities, this project could make significant contributions to fields such as automated software development, education, and real-time coding assistance. Ultimately, this work paves the way for a more efficient, accessible, and user-friendly approach to programming, with the potential to revolutionize how code is written and utilized across various domains.

# REFERENCES

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30.This paper introduced the Transformer model, which revolutionized NLP and served as the foundation for many subsequent models used in this project, including CodeGen.

[2] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neel, B., Shyam, P., Sastry, G., Askell, A., & co. (2020). Language Models are Few-Shot Learners. Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020).The paper introduced GPT-3, a model that influenced the development of AI-based code generation systems.

[3] Salesforce Research (2021). CodeGen: An Open-Source Code Generation Model. Salesforce Research.This paper details the CodeGen model used in the project, which is fine-tuned for code generation from natural language descriptions.

[4] Hugging Face, Inc. (2020). Transformers: State-of-the-Art Natural Language Processing. https://huggingface.co/docs/transformers/Hugging Face provides a comprehensive library for transformer models, which were crucial for tokenization, model inference, and code generation.

[5] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of NAACL-HLT 2019.BERT, another transformer-based model, inspired many code generation tools and helped shape the understanding of pre-trained language models.

[6] PyTorch (2016). PyTorch: An Imperative Style, High-Performance Deep Learning Library. https://pytorch.org/PyTorch was used as the underlying deep learning framework for training, fine-tuning, and running the transformer model for code generation.

[7] Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. Proceedings of the International Conference on Learning Representations (ICLR 2015).Adam was used as the optimizer for fine-tuning the model on specific tasks.

[8] Ruder, S. (2019). Neural Transfer Learning for Natural Language Processing. https://rpradeepmenon.medium.com/This article elaborates on the concepts of

transfer learning, which were used to fine-tune pre-trained models like CodeGen for specific tasks such as code generation.

[9] Vaswani, A., Bengio, S., & Bengio, Y. (2017). Attention is All You Need: A Survey of Transformer Models. Foundations and Trends in Machine Learning.This review paper gives an in-depth understanding of transformer models, which form the backbone of the project.

[10] Postman (2021). Postman API Platform. https://www.postman.com/Postman was used to test and debug the API endpoints during the development of the Flask application.