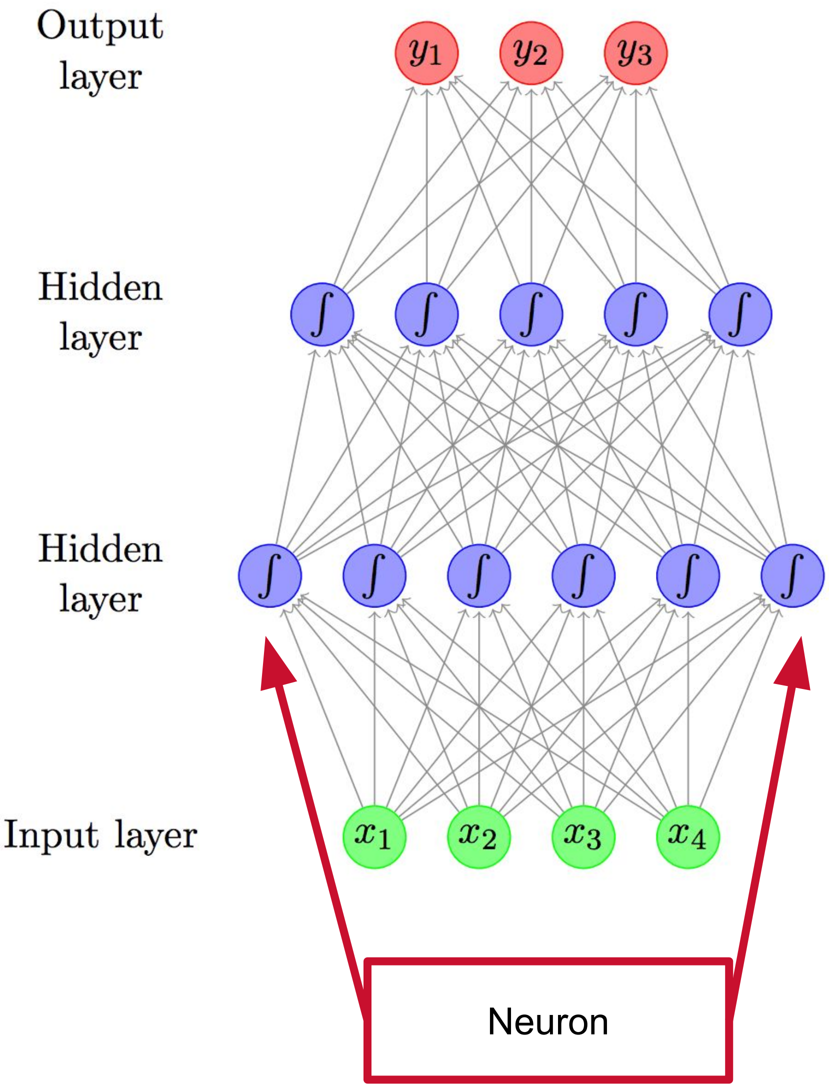
# Explainable AI For Source Code Applications

SDMAY25-30

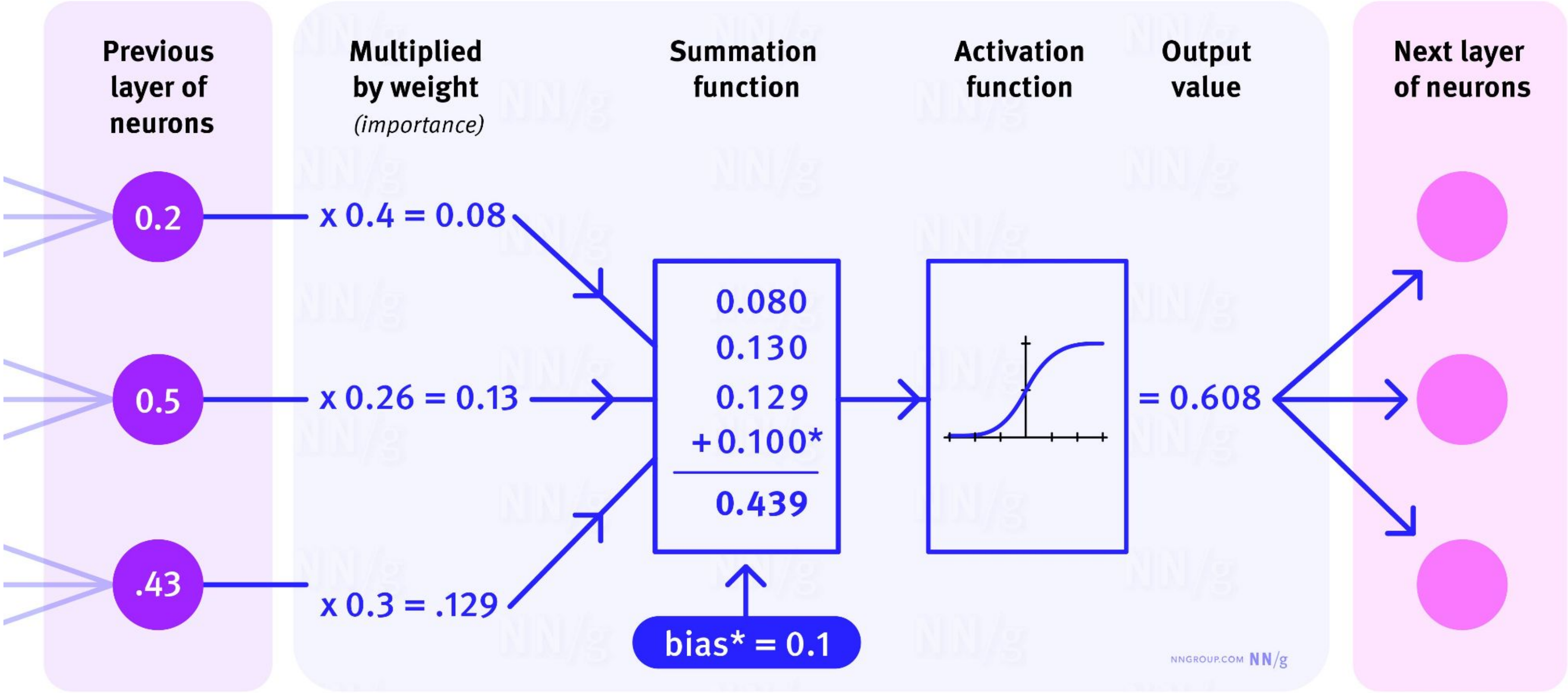
*Manjul Balayar, Kellan Bouwman, Sam Frost, Akhilesh Nevatia, Ethan Rogers*
*Client: Dr. Ali Jannesari/ISU SwAPP Lab*
*Advisor: Arushi Sharma*

Iowa State University

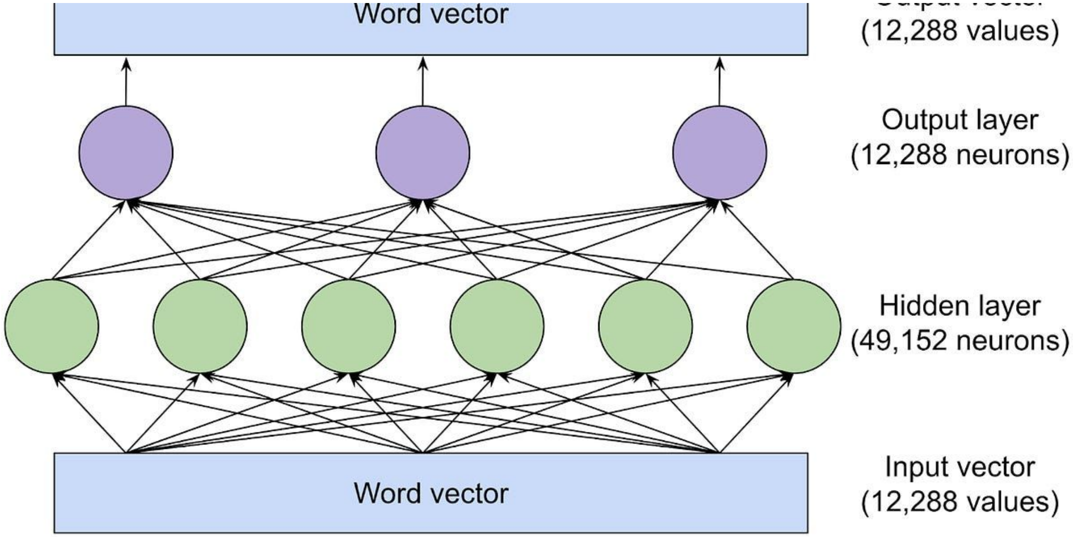
- Machine Learning / Statistical Learning:
  - Machine learning: A broad discipline that focuses on how computers can learn from data
  - Statistical learning: A branch of artificial intelligence that focuses on turning raw data into actionable information. Statistical learning theory is a framework that uses statistical and functional analysis to build models that can make predictions and draw conclusions from data
- Neuron:
  - A collection of a set of inputs, a set of weights, and an activation function.
  - It translates these inputs into a single output.
- Deep Learning:
  - Type of machine learning based on artificial neural networks in which multiple layers of processing are used to extract progressively higher level features from data

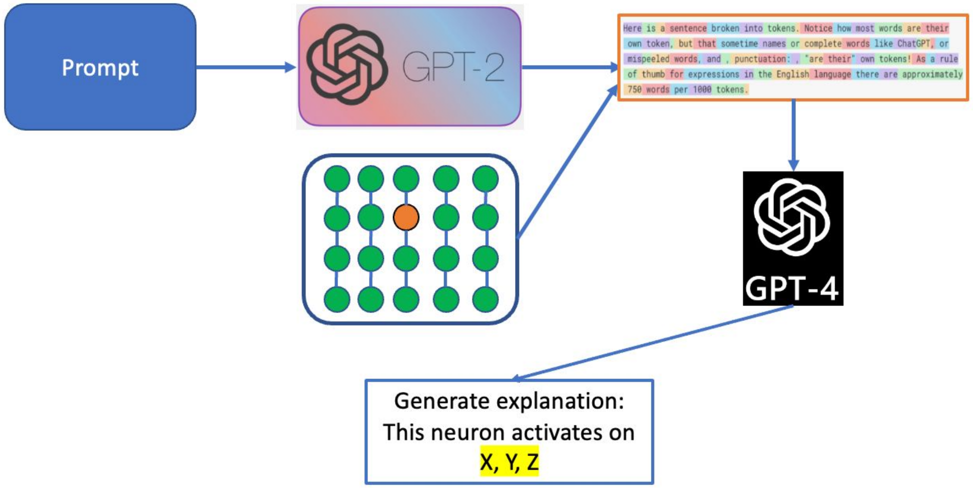


# How a Single Artificial Neuron Works



- LLM:
  - Large Language Model
  - Form of Deep learning on NLP (Natural Language Processing)
- Generative AI:
  - A type of AI that uses generative models to create new content, such as text, images, videos, music, and audio
  - Based off an LLM (example: ChatGPT3.5)
- Neuron Activation
  - Non-linear function that we apply over the input data coming to a particular neuron and the output from the function will be sent to the neurons present in the next layer as input
  - A Path of Neurons





Here is a sentence broken into tokens. Notice how most words are their own token, but that sometime names or complete words like ChatGPT, or mispeeled words, and , punctuation: , "are their" own tokens! As a rule of thumb for expressions in the English language there are approximately 750 words per 1000 tokens.

Generate explanation: This neuron activates on X, Y, Z

## Project Overview

- Client
    - Dr. Ali Jannesari/ISU SwAPP Lab
- Abstract
    - Focus on auto-labeling code datasets using AST tools, regular expressions, and LLM-generated labels.
- Goal
    - Deepen the understanding of LLMs and generative AI by analyzing neuron activations and applying metrics and heuristics. The project will also unify two existing code bases into a flexible and scalable framework that can be deployed seamlessly across Colab, local environments, and HPC clusters.

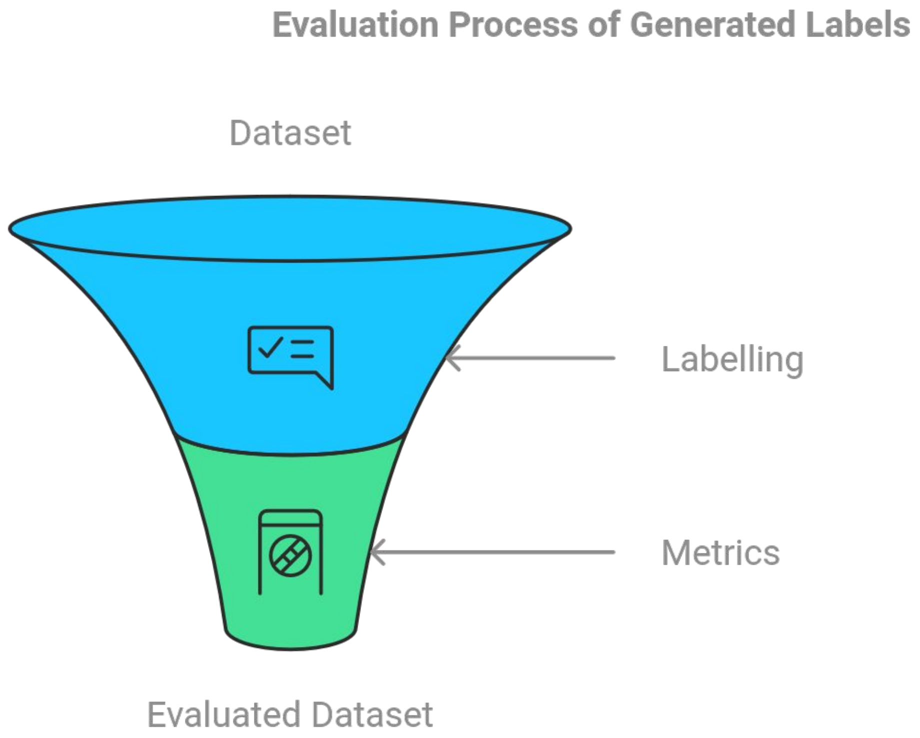
## Project Overview - Continued

- Users
    - Researchers
        - ML Engineers / Researchers
        - Prompt Engineers / Researchers
        - Computer Scientists
    - Students
        - Graduate Students
        - Undergraduate Students
    - Industry Professionals
- We aim to include students, researchers, and industry professionals as key users. Our documentation will prioritize accessibility and clarity for all experience levels, while the codebase will remain flexible and scalable to meet both academic and industry needs.

# Prototype - Overview

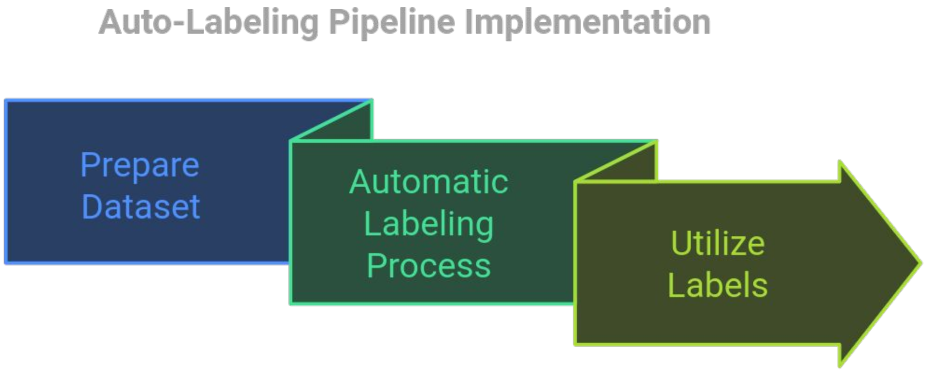
**Prototype: Clustering Module**

- Integrated initial versions of K-means and Agglomerative Clustering algorithms.
- Tested clustering on extracted activation data.



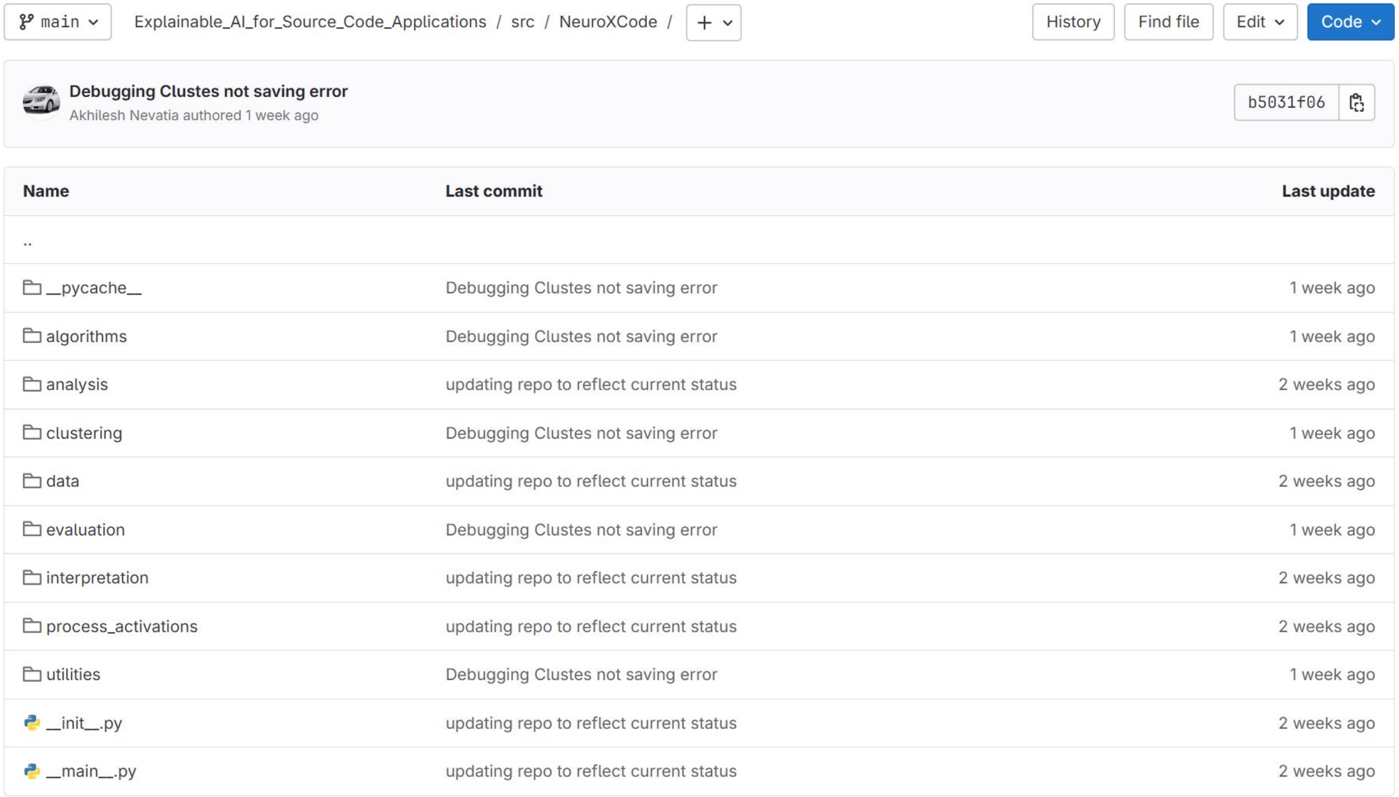
# Prototype - Overview

- **Purpose of Prototype**
  - Validate the feasibility of our modular design approach.
  - Identify challenges in integrating different components.
  - Gather initial performance metrics and user feedback.
- **Fit in Design Story**
  - Serve as foundational components for our latent concept analysis library.
  - Lay the groundwork for advanced features like automated labeling and visualization.
- **Learning Objectives**
  - Assess compatibility of technologies
  - Evaluate performance of clustering algorithms on large datasets.
  - Understand user needs for model selection and data input.


Auto-Labeling Pipeline Implementation
Prepare Dataset
Automatic Labeling Process
Utilize Labels

# Prototype - Demo (Code Run Through)

## Directory Structure



Explainable_AI_for_Source_Code_Applications / src / NeuroXCode /

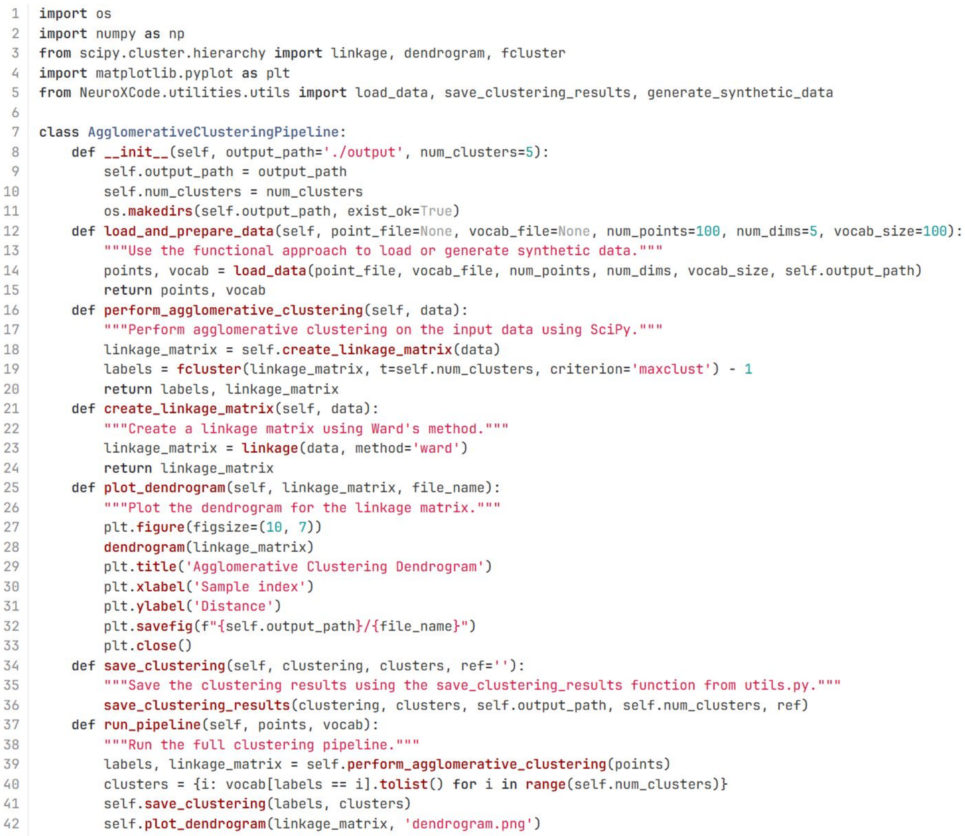
**Debugging Clustes not saving error**
Akhilesh Nevatia authored 1 week ago

b5031f06

| Name | Last commit | Last update |
|---|---|---|
| .. | | |
| __pycache__ | Debugging Clustes not saving error | 1 week ago |
| algorithms | Debugging Clustes not saving error | 1 week ago |
| analysis | updating repo to reflect current status | 2 weeks ago |
| clustering | Debugging Clustes not saving error | 1 week ago |
| data | updating repo to reflect current status | 2 weeks ago |
| evaluation | Debugging Clustes not saving error | 1 week ago |
| interpretation | updating repo to reflect current status | 2 weeks ago |
| process_activations | updating repo to reflect current status | 2 weeks ago |
| utilities | Debugging Clustes not saving error | 1 week ago |
| __init__.py | updating repo to reflect current status | 2 weeks ago |
| __main__.py | updating repo to reflect current status | 2 weeks ago |

# Prototype - Demo (Code Run Through)

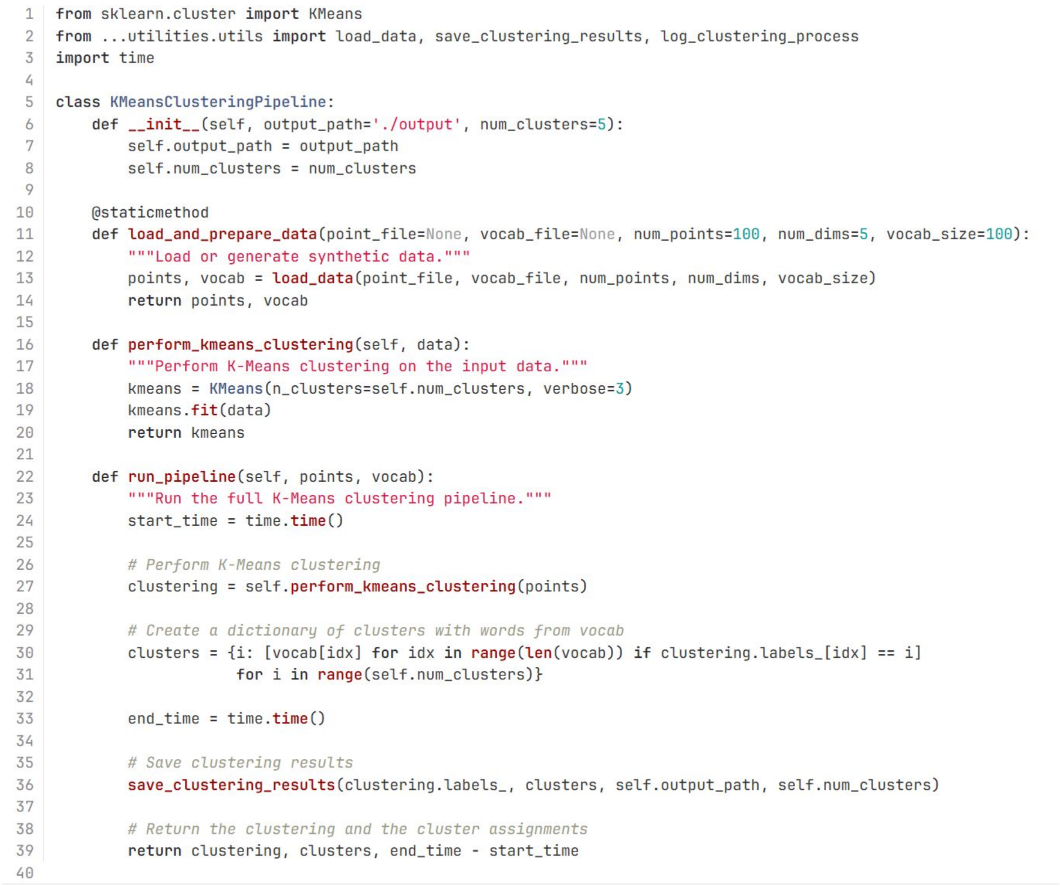
## Agglomerative Clustering



```python
import os
import numpy as np
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
import matplotlib.pyplot as plt
from NeuroXCode.utilities.utils import load_data, save_clustering_results, generate_synthetic_data

class AgglomerativeClusteringPipeline:
    def __init__(self, output_path='./output', num_clusters=5):
        self.output_path = output_path
        self.num_clusters = num_clusters
        os.makedirs(self.output_path, exist_ok=True)
    def load_and_prepare_data(self, point_file=None, vocab_file=None, num_points=100, num_dims=5, vocab_size=100):
        """Use the functional approach to load or generate synthetic data."""
        points, vocab = load_data(point_file, vocab_file, num_points, num_dims, vocab_size, self.output_path)
        return points, vocab
    def perform_agglomerative_clustering(self, data):
        """Perform agglomerative clustering on the input data using SciPy."""
        linkage_matrix = self.create_linkage_matrix(data)
        labels = fcluster(linkage_matrix, t=self.num_clusters, criterion='maxclust') - 1
        return labels, linkage_matrix
    def create_linkage_matrix(self, data):
        """Create a linkage matrix using Ward's method."""
        linkage_matrix = linkage(data, method='ward')
        return linkage_matrix
    def plot_dendrogram(self, linkage_matrix, file_name):
        """Plot the dendrogram for the linkage matrix."""
        plt.figure(figsize=(10, 7))
        dendrogram(linkage_matrix)
        plt.title('Agglomerative Clustering Dendrogram')
        plt.xlabel('Sample index')
        plt.ylabel('Distance')
        plt.savefig(f"{self.output_path}/{file_name}")
        plt.close()
    def save_clustering(self, clustering, clusters, ref=''):
        """Save the clustering results using the save_clustering_results function from utils.py."""
        save_clustering_results(clustering, clusters, self.output_path, self.num_clusters, ref)
    def run_pipeline(self, points, vocab):
        """Run the full clustering pipeline."""
        labels, linkage_matrix = self.perform_agglomerative_clustering(points)
        clusters = {i: vocab[labels == i].tolist() for i in range(self.num_clusters)}
        self.save_clustering(labels, clusters)
        self.plot_dendrogram(linkage_matrix, 'dendrogram.png')
```

# Prototype - Demo (Code Run Through)

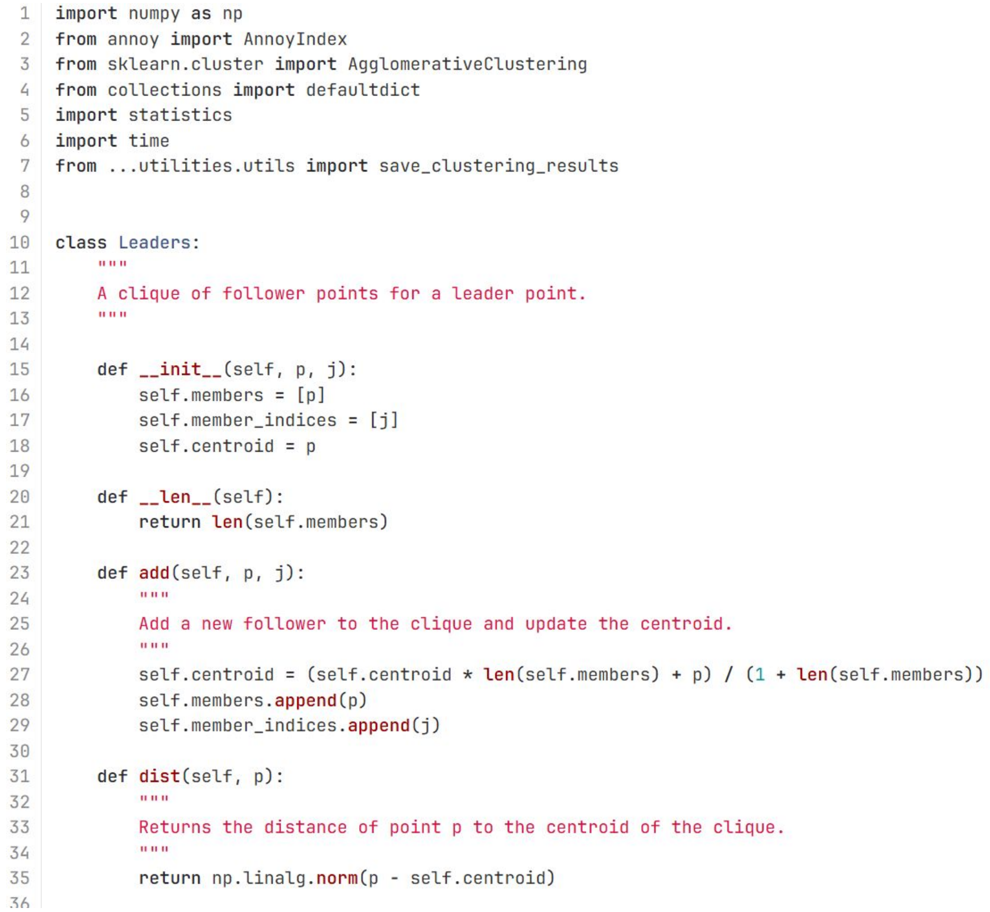
## K-Means Clustering



```python
from sklearn.cluster import KMeans
from ...utilities.utils import load_data, save_clustering_results, log_clustering_process
import time

class KMeansClusteringPipeline:
    def __init__(self, output_path='./output', num_clusters=5):
        self.output_path = output_path
        self.num_clusters = num_clusters

    @staticmethod
    def load_and_prepare_data(point_file=None, vocab_file=None, num_points=100, num_dims=5, vocab_size=100):
        """Load or generate synthetic data."""
        points, vocab = load_data(point_file, vocab_file, num_points, num_dims, vocab_size)
        return points, vocab

    def perform_kmeans_clustering(self, data):
        """Perform K-Means clustering on the input data."""
        kmeans = KMeans(n_clusters=self.num_clusters, verbose=3)
        kmeans.fit(data)
        return kmeans

    def run_pipeline(self, points, vocab):
        """Run the full K-Means clustering pipeline."""
        start_time = time.time()

        # Perform K-Means clustering
        clustering = self.perform_kmeans_clustering(points)

        # Create a dictionary of clusters with words from vocab
        clusters = {i: [vocab[idx] for idx in range(len(vocab)) if clustering.labels_[idx] == i]
                    for i in range(self.num_clusters)}

        end_time = time.time()

        # Save clustering results
        save_clustering_results(clustering.labels_, clusters, self.output_path, self.num_clusters)

        # Return the clustering and the cluster assignments
        return clustering, clusters, end_time - start_time

```

# Prototype - Demo (Code Run Through)

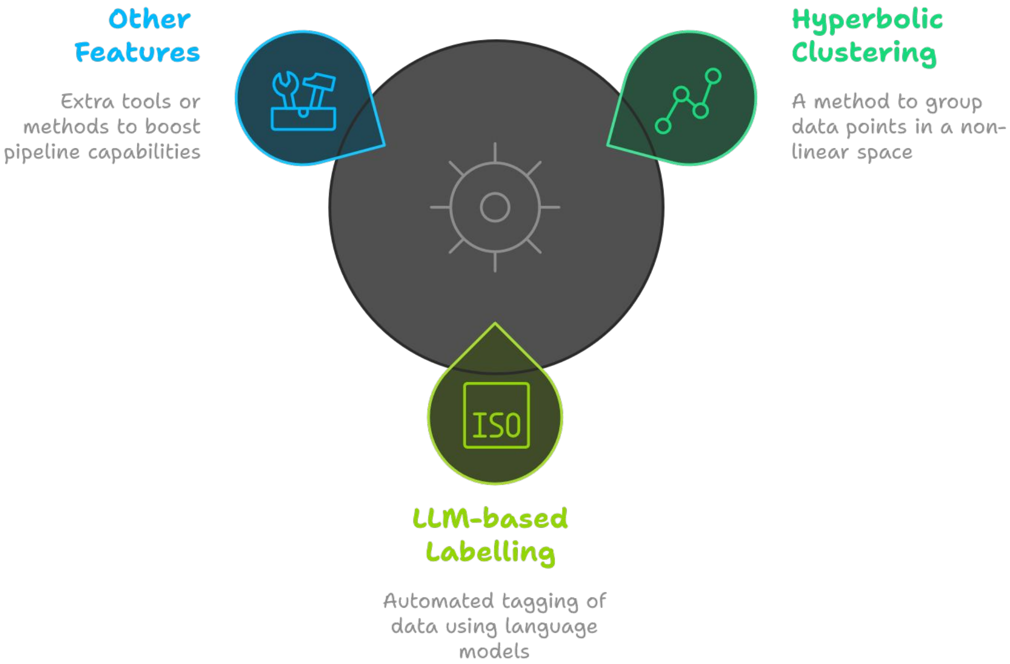
## Leaders Clustering



```python
import numpy as np
from annoy import AnnoyIndex
from sklearn.cluster import AgglomerativeClustering
from collections import defaultdict
import statistics
import time
from ...utilities.utils import save_clustering_results


class Leaders:
    """
    A clique of follower points for a leader point.
    """

    def __init__(self, p, j):
        self.members = [p]
        self.member_indices = [j]
        self.centroid = p

    def __len__(self):
        return len(self.members)

    def add(self, p, j):
        """
        Add a new follower to the clique and update the centroid.
        """
        self.centroid = (self.centroid * len(self.members) + p) / (1 + len(self.members))
        self.members.append(p)
        self.member_indices.append(j)

    def dist(self, p):
        """
        Returns the distance of point p to the centroid of the clique.
        """
        return np.linalg.norm(p - self.centroid)

```

# Prototype - Implications

- **Performance Optimization**
  - Enhance activation extraction efficiency via batch processing and parallelization.
  - Optimize clustering algorithms for handling high-dimensional data.
- **Future Development**
  - Integrate automated labeling using LLMs and DSPY2.
  - Develop the alignment and metrics evaluation module.
  - Expand analysis and visualization tools for deeper insights.



**Other Features**

Extra tools or methods to boost pipeline capabilities

**Hyperbolic Clustering**

A method to group data points in a non-linear space

**LLM-based Labelling**

Automated tagging of data using language models