**Group 2**

**Network Anomaly Detection**

Chandana Chevuturi 11664381

Naveen Ajay Karasu 11646981

Surya Simha Reddy Chintha 11702127

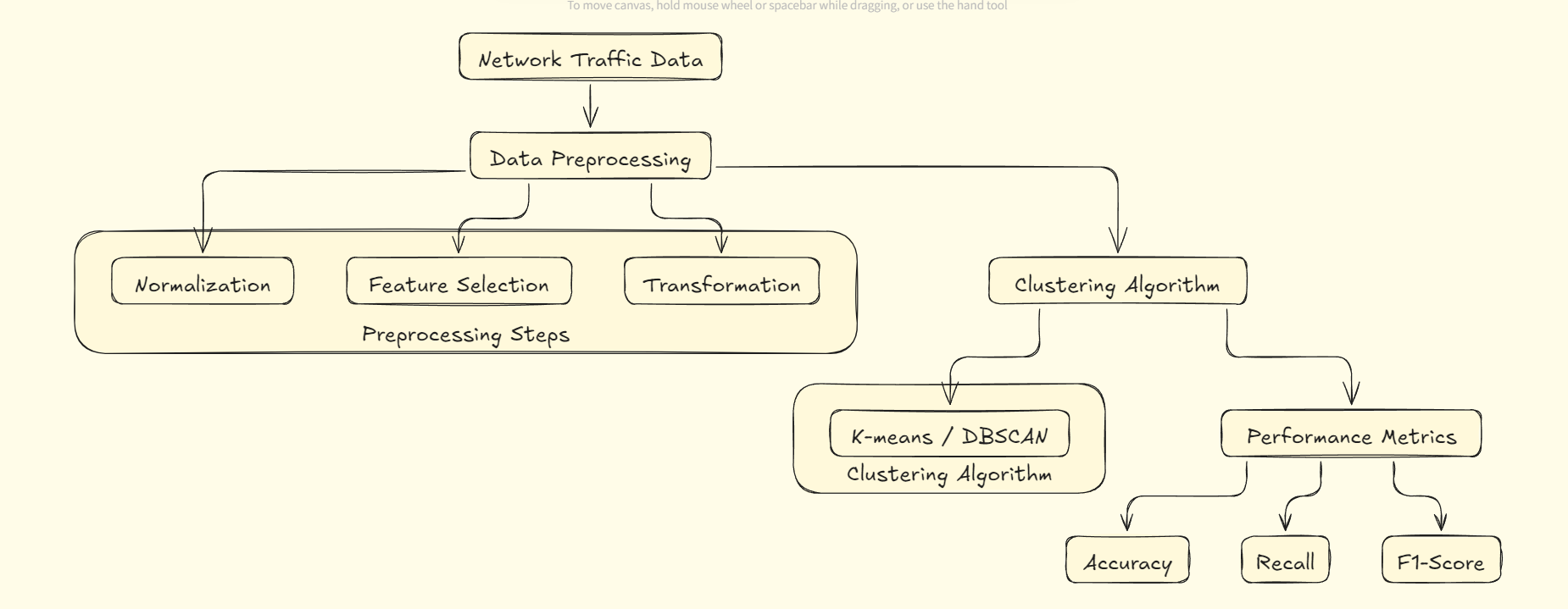
Pavan Kumar Reddy Bhumireddy 11656786

Sai Yashwanth Reddy Gujjula 11641520

Milestone First Report

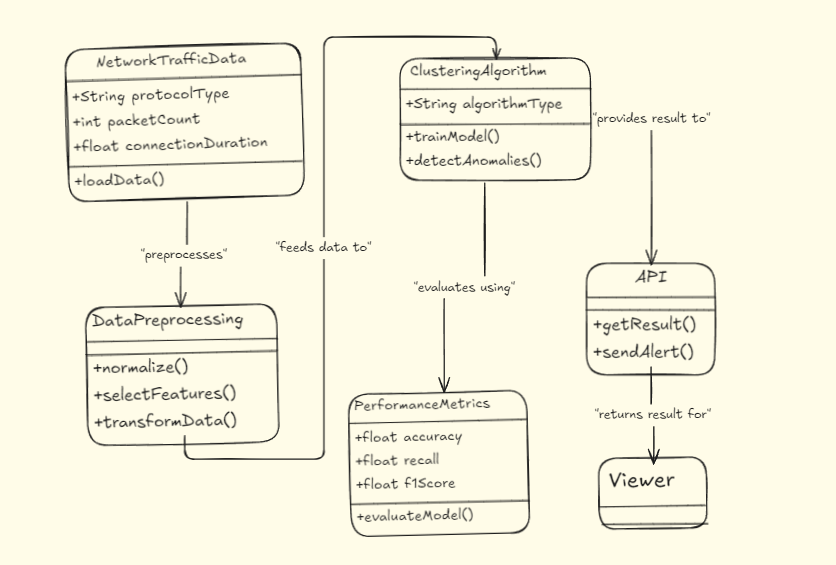
**System diagram:** The diagram represents the network anomaly detection system. We developed using machine learning techniques. It starts with the input of network traffic data, which includes various features like packet counts, protocol types and connection durations. The data is then pre-processed through data preprocessing, where normalization, feature selection, and transformation are applied to prepare the data for optimal machine learning performance.

After preprocessing, the data is passed to a clustering algorithm K-means to identify patterns and detect any anomalies in the network. The system's performance is then evaluated using key metrics like accuracy, recall, and the F1-score. Which then can be used to assess how well the model detects and classifies network anomalies.



**Class diagram** illustrates the architecture of our network anomaly detection system with an API feature for user interaction. The system starts with the NetworkTrafficData class, which loads data containing key features like protocol type, packet count, and connection duration. This data is then passed through the DataPreprocessing class, where it's normalized, important features are selected, and the data is transformed.

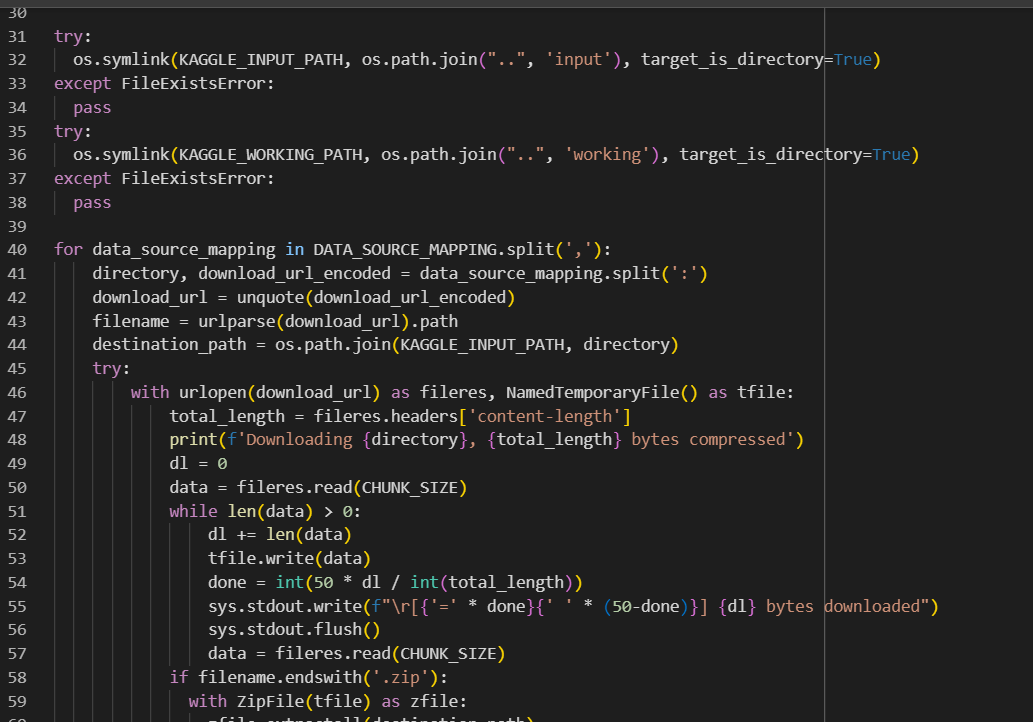
Once the data is preprocessed, it is sent to the ClusteringAlgorithm class, which is responsible for training the model and detecting anomalies using machine learning techniques. The results are then evaluated using the PerformanceMetrics class, which calculates accuracy, recall, and F1-score. Finally, the API class allows users to interact with the system by fetching the anomaly detection results and sending alerts providing a seamless way for viewers to access real-time insights from the model.

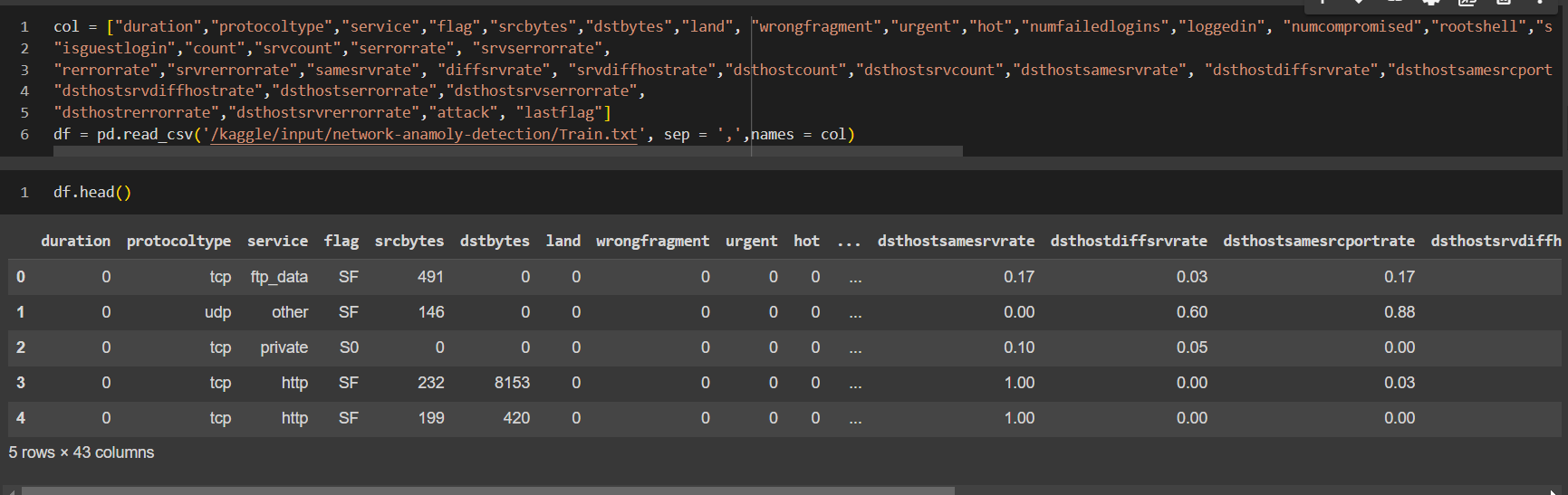


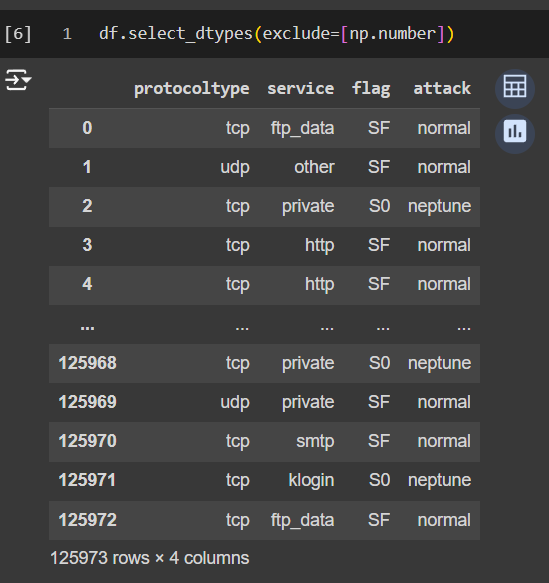
**Sample Code snippets**: The code follows a structured approach for analyzing network traffic data and applying machine learning techniques for anomaly detection.

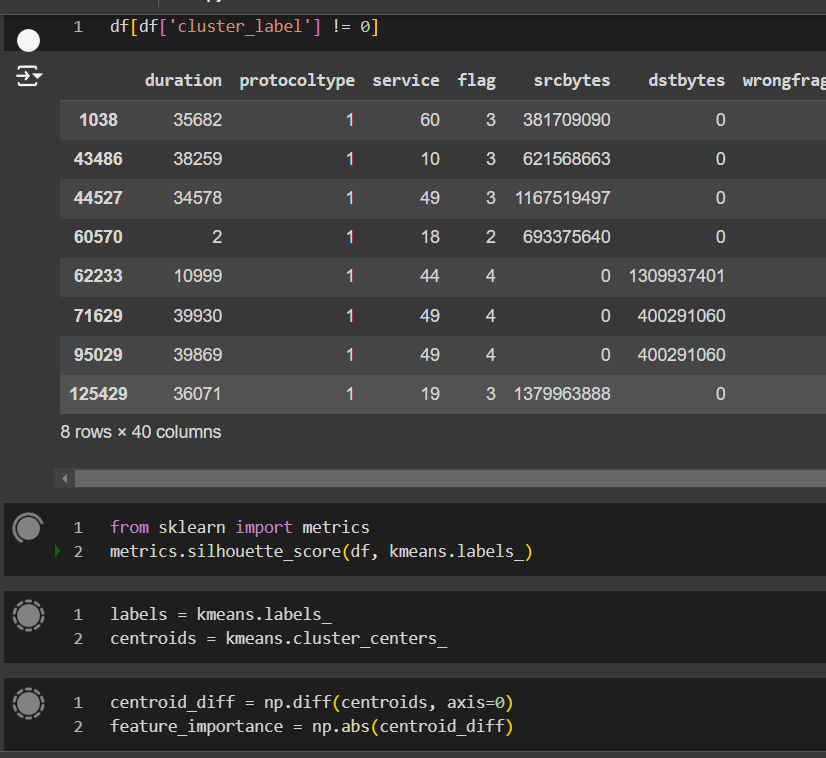
At first, the code handles the loading and initial inspection of the dataset. The dataset is read from a CSV file, which contains multiple features representing network traffic attributes, such as duration, protocoltype, service, and many more. After loading the dataset, it is displayed using the df.head() function to provide an overview of the first few rows and ensure that the dataset has been imported correctly. The dataset consists of 43 columns, where each representing a different attributes of the network traffic. Which will later be used for anomaly detection.

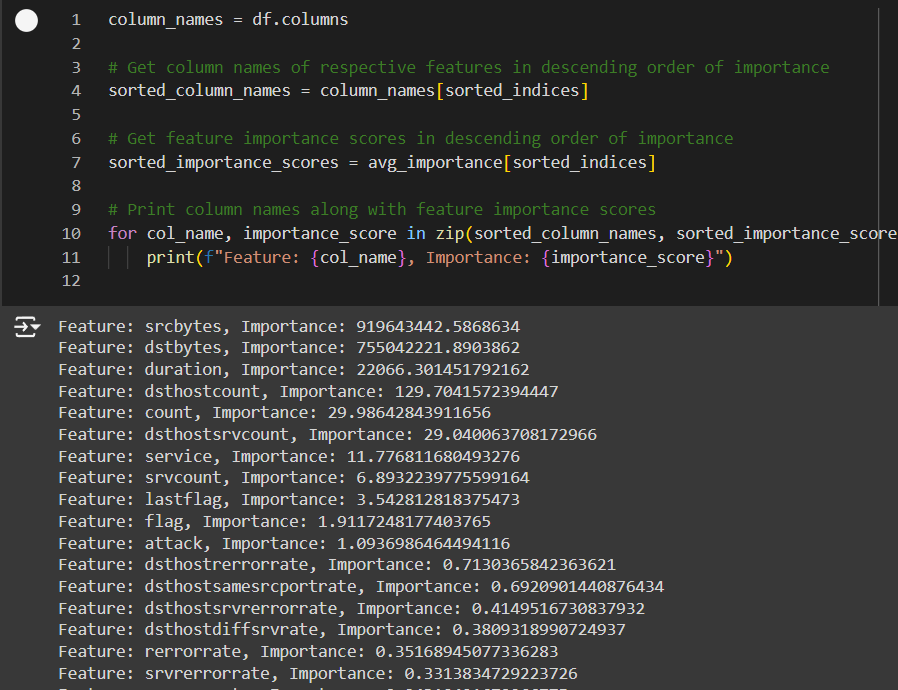
Next, in the code we focus on feature importance and clustering using the KMeans algorithm. Initially, the KMeans model assigns a cluster label to each record. The dataset is then filtered to display only the rows where the cluster label is not zero (df['cluster\_label'] != 0). This selection likely represents anomalous data points that belong to a specific cluster, which is need for us to do the anomaly detection task. The silhouette score is calculated after that, which can be used has an evaluation metric to assess how well the clustering algorithm has performed. This metric is useful for determining the compactness and separation of clusters in the data.

After that feature importance is calculated by comparing the centroids of the clusters. The code calculates the absolute differences between the centroids (np.abs(centroid\_diff)) and sorts the features in descending order of their importance. This helps to identify which features contribute the most to the clustering results. The code then prints out each feature along with its calculated importance score, allowing for better understanding and interpretation of the most influential and important features in the context of network anomaly detection.  










We currently working on designing the model and once we have completed that, we will start preparing the UI and Pki file to integrate them to create an API.