Predicting Customer Churn in Telecommunications Industry using Bayesian Networks and Classification

Hamzah Asghar Farooqi (13191038)

Muhammad Ahmad Saeed Chaudhry (13332295)

Faculty of Engineering Environment and Computing

Coventry University

farooqih@uni.coventry.ac.uk

Abstract: *In the highly competitive telecommunications industry, predicting customer churn is crucial for retaining customers and ensuring business success. In the paper, we apply advanced ML techniques, including Bayesian Networks and Classification, to predict customer churn based on customer demographics, usage patterns, and customer service experience The Telco Customer Churn dataset is used in the paper to preprocess data and assess the performance of different machine learning algorithms for predicting customer churn. The study concludes that Bayesian Networks and Gaussian process classification are successful in predicting churn and can aid telecommunication companies in retaining customers. This research showcases the potential of machine learning in addressing real-world issues and provides valuable understanding of the intricate correlation between customer behavior and churn.*

Keywords*, customer churn, machine learning techniques, Bayesian Networks, Classification, customer demographics, usage patterns, customer service experience, performance, Gaussian process classification, proactive measures, real-world problems.*

1. Introduction

The telecommunications industry is highly competitive, and retaining customers is crucial for the success of telecommunication companies. Customer churn, or the rate at which customers discontinue their services, is a major concern for the industry In years, ML techniques have shown great potential in predicting customer churn and helping companies take proactive measures to retain their customers. In this project, we aim to use advanced machine learning techniques, such as Bayesian Networks and Classification, to predict customer churn based on several input variables, such as customer demographics, usage patterns, and customer service experience. The performance of various machine learning algorithms is evaluated using the Telco Customer Churn dataset to determine the most effective approach for predicting customer churn. This study adds to the growing research on the application of machine learning in the telecommunications industry and showcases its potential for solving real-world problems.

Figure 1: Customer Churn (picture taken from nddata.com)

Evaluating customer churn is a crucial aspect for the growth of any enterprise, even though it may not be the most pleasant metric to track. It provides a realistic understanding of the company's customer retention rate. Determining success requires measuring failures as well. While aiming for 100% customer retention is an ideal goal, it is not practical. To calculate churn rate, the number of lost customers (20) is divided by the initial number of customers (400), resulting in a churn rate of 5% in the given scenario. Although the goal is to achieve a churn rate close to 0%, it is vital for the company to regularly monitor and prioritize this metric. It's worth noting that there are various methods for calculating churn rate, and companies should choose the one that suits their specific needs.

The telecommunications industry is witnessing a surge in the mobile market, which has become the fastest-growing segment. Globally, over 75% of phone calls are made using mobile phones, and customer retention has become a top priority for companies in competitive markets with low switching costs. The telecommunications industry is among the sectors that are most affected by high customer churn rates, which can reach up to 30% annually. To prevent customer churning, mobile telecommunications companies must identify customers at risk of churning before it happens. Hence, implementing an accurate predictive model that can forecast potential churners has become crucial. The predictive model must accurately identify customers who are likely to churn soon, enabling companies to take proactive measures to retain them. This helps to avoid wasting resources on customers who would have remained loyal anyway. However, identifying only those customers who are likely to churn can be a complex task, especially in the case of prepaid mobile telecommunications, where no contractual obligation exists. Therefore, developing an accurate predictive model that can identify potential churners in such scenarios requires careful consideration and expertise.

Recent advancements in algorithm development have made Bayesian networks a leading area of research in artificial intelligence. The transmission of probabilistic information through graphs has been made possible by these algorithms, and it has become apparent to the machine learning community that the probabilistic nature of Bayesian networks enables learning from data. Initially, researchers focused on identifying conditional independence structures in data and creating Bayesian networks. However, new Bayesian methods were soon introduced for learning from data, and these methods were subsequently refined in later research.

2. Related Work

- **Customer churn prediction in telecom using machine learning in big data platform.**

The telecom industry faces a significant problem with customer churn, and companies attempt to predict customer behavior to prevent it. This work uses machine learning techniques to develop a churn prediction model that achieved a high-performance measure of 93.3% AUC. The model uses social network analysis (SNA) features to improve predictions by considering customer connections. The model was tested on a large dataset from a telecom company, providing information on customer behavior for nine months. The model was trained, tested, and evaluated on this dataset using four different algorithms, with the XGBOOST algorithm proving to be the most effective for classifying potential churners.

- **Predicting Customer Churn in Telecom Industry using Multilayer Perceptron Neural Networks: Modeling and Analysis**

The loss of customers to competitors, known as churn, can lead to significant financial loss for businesses. As a result, many companies are exploring various techniques to predict churn rates and create effective customer retention plans. In this study, the prediction of customer churn in a telecommunications company in Jordan is investigated using Multilayer Perceptron (MLP) neural networks with back-propagation learning. The study examines different MLP topologies to create churn classification models and compares two MLP-based approaches to determine the key factors that impact churn rates. Real customer data is utilized for the analysis.

3. Bayesian Networks and Classification

Bayesian networks utilize a directed acyclic graph (DAG) to depict the probabilistic connections among a set of random variables X, including X1, X2, Xp. Each node, represented by vi, refers to a specific random variable Xi. The graphical separation, or absence of an edge, in the DAG corresponds to probabilistic independence between the variables, denoted as ⫫G. This relationship between graphical separation and independence is known as the independency map. An independence map is a graphical representation of the probabilistic dependence structure P of a set of random variables X, denoted by a graph G = (V, A), where there is a one-to-one correspondence between the variables in X and the nodes in V.

For disjoint subsets A, B, and C, the independency relationship is represented as

$A \perp\!\!\!\perp_p B \mid C \Leftarrow A \perp\!\!\!\perp_g B \mid C$ (1).

$A \perp\!\!\!\perp_p B \mid C \Rightarrow A \perp\!\!\!\perp_g B \mid C$ (2)

A directed acyclic graph (DAG) G that serves as both a directed map (D-map) and an independence map (I-map) represents the probabilistic dependence structure P perfectly, and P is considered faithful to G. The d-separation criterion explains the relationship between G and conditional independence.

The Markov blanket of a node A in the set V is defined as the smallest subset S of V such that A is probabilistically independent from the rest of the variables in V, given the variables in S.

$A \perp\!\!\!\perp_P V - S - A \mid C$ (3)

In a Bayesian network, the Markov blanket of a node A consists of its predecessors, successors, and successors' predecessors, as illustrated in Figure 2.

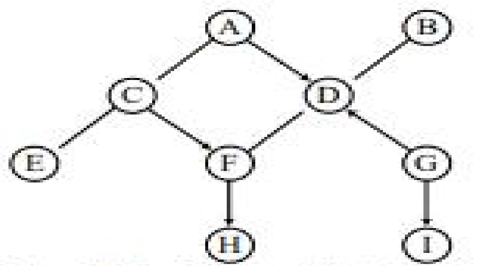
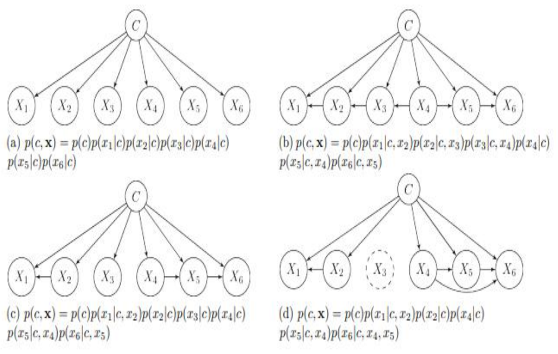


Figure 2:The Markov blanket of node A to G

The process of constructing a Bayesian network is known as learning, which includes two key phases: structure learning and parameter learning. Structure learning aims to identify the graph structure as the minimum independence map (I-map) of the dependence structure or an approximation of the probability distribution in the probability space.

The article describes different network structures produced by learning algorithms, which include Naive Bayes (NB), (ODE) like (TAN), Forest-augmented NB (FAN), (k-DB), Semi-naive Bayes (SNB), and averaged one-dependence estimators (AODE).

The bnclassify package creates augmented naive Bayes models, as shown in Figure 3. These models include the naive Bayes (NB) model, which assumes feature independence given the class, and the One-dependence estimator (ODE) model, which allows each predictor to depend on at most one other predictor. The Tree-augmented naive Bayes (TAN)


C
X_1
X_2
X_3
X_4
X_5
X_6
(a) $p(c,\mathbf{x}) = p(c)p(x_1|c)p(x_2|c)p(x_3|c)p(x_4|c)$ $p(x_5|c)p(x_6|c)$
(b) $p(c,\mathbf{x}) = p(c)p(x_1|c,x_2)p(x_2|c,x_3)p(x_3|c,x_4)p(x_4|c)$ $p(x_5|c,x_4)p(x_6|c,x_5)$
(c) $p(c,\mathbf{x}) = p(c)p(x_1|c,x_2)p(x_2|c)p(x_3|c)p(x_4|c)$ $p(x_5|c,x_4)p(x_6|c,x_5)$
(d) $p(c,\mathbf{x}) = p(c)p(x_1|c,x_2)p(x_2|c)p(x_4|c)$ $p(x_5|c,x_4)p(x_6|c,x_4,x_5)$
Figure 3: taken from Semantic Scholar

model is a special case of ODE with n - 1 augmenting arcs, while Forest-augmented naive Bayes (FAN) may have fewer than n - 1.

4. Problem and Dataset Description

The telecommunications industry is characterized by fierce competition, where multiple providers offer similar services to consumers. This competition is not only about attracting new customers but also about retaining existing ones. For telecommunications companies, retaining existing customers is crucial for long-term success because acquiring new customers tends to be more expensive. Therefore, customer churn, which refers to the rate at which customers terminate their services, is a significant concern in the industry. Churn can occur for various reasons, including price, quality of service, and customer experience. Understanding the factors that lead to churn and predicting which customers are most likely to leave is critical for telecommunications companies to take proactive measures to retain their customers.

In this project, we aim to use advanced machine learning techniques, specifically Bayesian Networks and Classification, to predict customer churn based on several input variables, such as customer demographics, usage patterns, and customer service experience. By analyzing these variables, we can identify patterns that are indicative of customers who are at risk of churning. This information can help telecommunication companies take proactive measures to retain customers, such as offering promotions or improving customer service. Our goal is to develop a model that can accurately predict customer churn and provide insights into the factors that contribute to churn.

The project will be using the Telco Customer Churn dataset which is publicly available on Kaggle. The dataset consists of information related to customer demographics, usage patterns, and customer service experience. The dataset is organized for example every row corresponds to a customer, and every column has an attribute of the customer, as specified by the column metadata. The dataset comprises 7043 rows, each row representing a unique customer, and 21 columns representing various attributes of the customer described in the column metadata. The target variable of interest is the "Churn" column. The primary objective of this project is to use ML techniques to accurately predict in the telecommunications industry. We explain some of the attributes in table 1.

Table 1: Telco Customer Churn attributes

Attribute	**Name Description**
customerID	Customer ID
gender	The customer male or female
SeniorCitizen	customer is senior citizen or not (1, 0)
Partner	Whether the customer has a partner or not (Yes, No)
Dependents	Whether the customer has dependents or not (Yes, No)
tenure	Number of months the customer has stayed with the company
PhoneService	Whether the customer has a phone service or not (Yes, No)
MultipleLines	Whether the customer has multiple lines or not (Yes, No, No phone service)

InternetService	Customer’s internet service provider (DSL, Fiber optic, No)
OnlineSecurity	Whether the customer has online security or not (Yes, No, No internet service)

Before using the dataset, we will perform data cleaning by removing any outliers or missing values. Furthermore, we will perform feature engineering to identify the most relevant features that can help us predict customer churn accurately. By analyzing and understanding the dataset, we will be able to develop a model that can assist telecommunication companies in taking proactive measures to retain their customers.

5. Methods

Numerous machine learning techniques can be applied to predict customer churn in the telecommunications sector. In our paper, we employed logistic and random forest regression to address this challenge. Logistic regression is a widely used and uncomplicated analytical technique that explores the relationship between a categorical variable and one or more categorical or continuous variables. In logistic regression, the probability of a target label (y) given a set of independent variables (xi) is estimated using the equation P(y=1|xi) = p^, where xi represents the independent variables, yi corresponds to the target label, and Y = {−1,+1}.

$$\hat{p} = \frac{1}{1 + \exp(-\sum_{j=0}^{n} \beta_i x_{i_j})}$$

The regression equation consists of an intercept β0 and regression coefficients βj, j= {1, 2,..,n}.

$$\sum_{j=0}^{n} \beta_j x_{i_j} = \beta_0 + \beta_1 x_{i_1} + \cdots + \beta_n x_{i_n} = \ln(\hat{p}/1 - \hat{p}) = \text{logit}\,(\hat{p})$$

It is a commonly used statistical method for binary classification problems. We can use logistic regression to model the relationship between customer features and the likelihood of churn.

Random Forest: It is an ensemble method that can be used for classification problems. Random forest can handle non-linear relationships and can capture feature interactions, which makes it a useful method for customer churn prediction. Random forest is an ensemble method that operates on decision trees by using the technique of bootstrap aggregation (bagging) across multiple trees. The decision tree is trained on multiple datasets drawn from the original dataset with replacement, and several decision trees are trained. The result is determined by calculating the average outcomes from each tree.

SVM: It is a technique for binary organization that tries to find a hyperplane that separates the classes.

Naive Bayes: This is a probabilistic method that is used for classification problems. Naive Bayes is simple and efficient, making it a good option when dealing with large datasets.

Neural Networks: Inspired by the structure and function of the human brain, they belong to a class of machine learning models. NN can capture complex relationships between features and the target variable, making them a powerful method for customer churn prediction.

Bayesian Networks: It is a probabilistic graphical model that can capture complex relationships between variables. Bayesian networks can be useful for predicting customer churn as they can handle uncertainty and missing data, which are common in real-world datasets.

6. Experimental setup

The objective is to develop customized customer retention strategies by forecasting customer behavior using a dataset that comprises customer attributes like service subscriptions, tenure, payment method, billing preference, monthly and total charges, and demographic details such as age range, gender, and the presence of partners or dependents. The dataset also contains information about customers who left within the past month, indicated by the "Churn" column.

7. Data Pre-processing:

We first checked for any missing values in the dataset and found that there are eleven lost values. As this was a small percentage of the overall data, we decided to remove these rows from the dataset. Then We checked for outliers in the numerical columns using box plots and found that there were no significant outliers in the dataset The dataset comprised various categorical variables such as gender, partner, dependents, phone service, multiple lines, internet service, online security, online backup, device protection, tech support, streaming TV, and streaming movies.. We converted these categorical variables into dummy variables using one-hot encoding. The dataset was slightly imbalanced, with 26.5% of customers churning and 73.5% of customers not churning. To address this, we used a stratified sampling technique while splitting the data into training and testing sets. Finally, we scaled the numerical features in the dataset using the StandardScaler() function from the scikit-learn library to ensure that all features were on the same scale.

Now we discuss the code. At first, we imported and installed several libraries including pgmpy, pandas, networkx, matplotlib.pyplot, sklearn, and seaborn. It also imports various classes and functions from these libraries. These libraries are used for probabilistic graphical modeling, data manipulation and analysis, graph creation and manipulation, visualization, model evaluation, and random forest regression. The code is preparing for the implementation of a machine learning model for predicting customer churn in the telecommunications industry using Bayesian Networks and Random Forest Regression.

```
# Load Telco customer churn dataset                                                    (1)
telco = pd.read_csv("/content/telecom_data.csv")
# Remove rows with missing values
telco = telco.dropna()
# Randomly select 8 variables
selected_vars = ["SeniorCitizen", "Partner", "Dependents", "PhoneService",  "InternetService", "Contract", "Pa
perlessBilling", "PaymentMethod"]
telco = telco[selected_vars + ["Churn"]]
```

From the above code 1, the code loads and prepares the Telco customer churn dataset by removing missing values and selecting a subset of relevant variables that will be used to predict customer churn.

```
# Convert all variables to categorical type                                             (2)
for col in telco.columns:
  telco[col] = pd.Categorical(telco[col])
# Randomly split data into training and testing sets
```

```
train, test = train_test_split(telco, test_size=0.3, random_state=123)
```

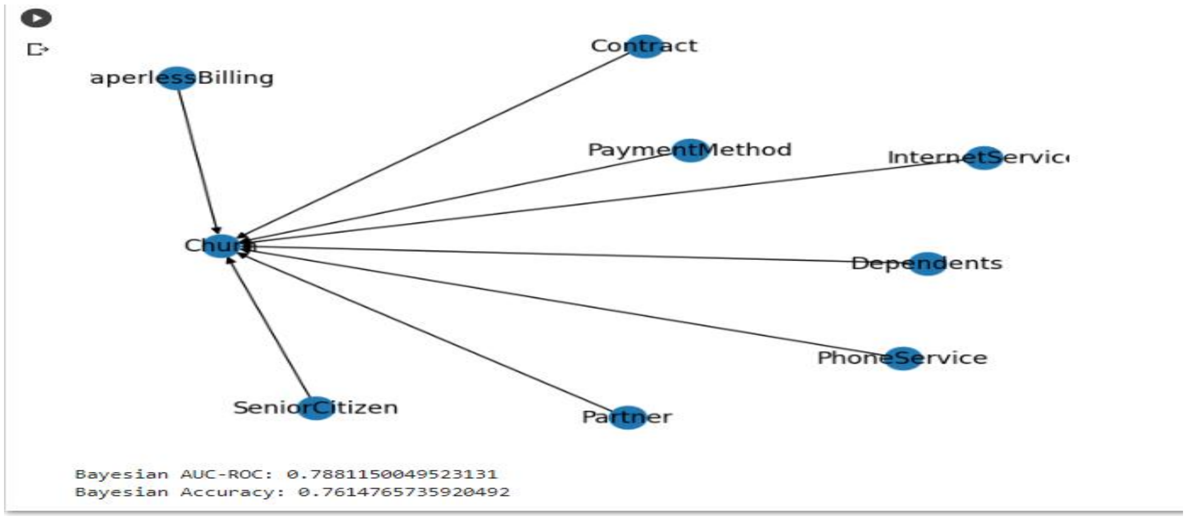
From the above code 2, we convert all variables to categorical type and splits the data randomly into training and testing sets. The function used for this purpose is train_test_split from the sklearn library, which creates a training set and a testing set with a 70:30 ratio, respectively. To ensure reproducibility, the random_state parameter is set to 123. The resulting training and testing sets are stored in variables named "train" and "test," respectively. Overall, this code is crucial for preparing the data for model building by converting variables to categorical type and splitting the data into training and testing sets.

```
# Create an empty Bayesian network structure

bn_model = BayesianNetwork()                                                    (3)

# Specify the node labels and the structure of the Bayesian network

for var in selected_vars:

   bn_model.add_node(var)

bn_model.add_node("Churn")

bn_model.add_edges_from([(var, "Churn") for var in selected_vars])

# Train the Bayesian network using the training data

bn_model.fit(train, estimator=MaximumLikelihoodEstimator)
```

From the above code 3 we creates and trains a Bayesian network on the selected variables in the Telco customer churn dataset using the pgmpy library.First, an empty Bayesian network structure is created using the BayesianNetwork() function and assigned to the variable bn_model. Next, the node labels and the structure of the Bayesian network are specified. A for loop is used to iterate over each selected variable and add it as a node to the network using the add_node() function. The "Churn" node is also added. The add_edges_from() function is used to add edges between each variable node and the "Churn" node. Overall, this code creates and trains a Bayesian network on the selected variables in the Telco customer churn dataset, which can be used for predicting customer churn.

```
# Draw the Bayesian network                                                     (4)

graph = nx.DiGraph(bn_model.edges())

nx.draw(graph, with_labels=True)

plt.show()

# Predict using the Bayesian network

infer = VariableElimination(bn_model)

test_prob_bn = []

for index, row in test.iterrows():

   q = infer.query(variables=["Churn"], evidence=row[selected_vars].to_dict())

   pred_prob = q.values[1]

   test_prob_bn.append(pred_prob)
```

```
# Evaluate the performance of the Bayesian model using AUC-ROC and accuracy

label_map = {"Yes": 1, "No": 0}

test["Churn"] = test["Churn"].map(label_map)

test_pred_bn = pd.Series(test_prob_bn, name="Churn")

test_pred_bn = test_pred_bn.map(lambda x: 1 if x > 0.5 else 0)


auc_bn = roc_auc_score(test["Churn"], test_prob_bn)

acc_bn = accuracy_score(test["Churn"], test_pred_bn)

print("Bayesian AUC-ROC:", auc_bn)

print("Bayesian Accuracy:", acc_bn)
```



```
Bayesian AUC-ROC: 0.7881150049523131
Bayesian Accuracy: 0.7614765735920492
```

Figure 4: Plotting of Bayesian Network

From the above code 4 and figure 4, we draws the Bayesian network, makes predictions using the trained model, and evaluates its performance. The first part draws the graph using the edges in the bn_model object. The second part uses the trained Bayesian network to predict churn for each row in the test data, and the predicted probabilities are stored in a list. In the third part, the model's performance is evaluated using AUC-ROC and accuracy metrics. The predicted probabilities are converted to a pandas series and mapped to 1 or 0 based on a threshold of 0.5. The roc_auc_score() and accuracy_score() functions from sklearn are used to compute the metrics, and the results are Bayesian AUC-ROC= 0.7881150049523131and Bayesian Accuracy: 0.7614765735920492

```
# Convert categorical variables to one-hot encoding                                   (5)

train = pd.get_dummies(train, columns=selected_vars)

test = pd.get_dummies(test, columns=selected_vars)

train["Churn"] = train["Churn"].map(label_map)
```

```
# Create a list of all the feature columns

feature_cols = list(train.columns)

feature_cols.remove("Churn")

# Train a Random Forest Regression model using the training data

rf_model = RandomForestRegressor(random_state=123)

rf_model.fit(train[feature_cols], train["Churn"]

# Predict using the Random Forest Regression model

test_pred_rf = rf_model.predict(test[feature_cols])

test_pred_rf = pd.Series(test_pred_rf, name="Churn")

# Evaluate the performance of the Random Forest Regression model using AUC-ROC and accuracy

test_pred_rf = test_pred_rf.apply(lambda x: 1 if x > 0.5 else 0)

auc_rf = roc_auc_score(test["Churn"], test_pred_rf)

acc_rf = accuracy_score(test["Churn"], test_pred_rf)

print("Random Forest Regression AUC-ROC:", auc_rf)

print("Random Forest Regression Accuracy:", acc_rf)
```

From the above code 5, This code performs binary classification using a RFR model. First, the categorical variables in the "train" and "test" data frames are converted to one-hot encoding using the get_dummies() function from pandas. The "Churn" column in the "train" data frame is then mapped to 1/0 using the label_map dictionary. A list of all the feature columns is created, excluding the "Churn" column. A RF Regression model is then trained using the eye columns & "Churn" column in "train" data frame. The trained model is used to predict the "Churn" column in the "test" data frame, and the predicted probabilities are converted to 1/0 based on a threshold of 0.5. Finally, the AUC-ROC and accuracy of RFR model are evaluated using the roc_auc_score() and accuracy_score() functions from the sklearn library, respectively. The results are Random Forest Regression AUC-ROC= 0.6704524967448279 and Random Forest Regression Accuracy =0.7576904874585897. The Byesian AUC-ROC is higher then Random forest Regression AUC-ROC and the Byesian Accuracy is also higher then Random forest Regression Accuracy. # Compare the performance of the two models

```
fpr_bn, tpr_bn, _ = roc_curve(test["Churn"], test_prob_bn)                                    (6)

fpr_rf, tpr_rf, _ = roc_curve(test["Churn"], test_pred_rf)

plt.plot(fpr_bn, tpr_bn, label="Bayesian Network")

plt.plot(fpr_rf, tpr_rf, label="Random Forest Regression")

plt.xlabel("False Positive Rate")

plt.ylabel("True Positive Rate")

plt.legend()
plt.show()

# Plot feature importance of the Random Forest Regression model

feat_importances = pd.Series(rf_model.feature_importances_, index=feature_cols)

feat_importances.sort_values().plot(kind="barh")

plt.xlabel("Feature Importance")

plt.title("Random Forest Regression Feature Importance")

plt.show()

# Plot the confusion matrix for the Random Forest Regression model

conf_mat = confusion_matrix(test["Churn"], test_pred_rf)

sns.heatmap(conf_mat, annot=True, fmt="d", cmap="Blues")

plt.xlabel("Predicted Label")

plt.ylabel("True Label")

plt.title("Random Forest Regression Confusion Matrix")

plt.show()


# Get the confusion matrix for the Bayesian model

cm_bn = confusion_matrix(test["Churn"], test_pred_bn)

# Plot the confusion matrix using seaborn heatmap

sns.heatmap(cm_bn, annot=True, fmt="d", cmap="Blues")

plt.title("Bayesian Model Confusion Matrix")

plt.xlabel("Predicted Label")

plt.ylabel("True Label")

plt.show()
```

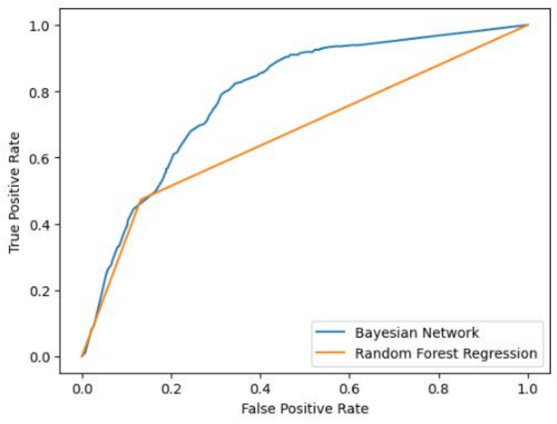


Figure 5: Plot of Bayesian and Random Forest Network

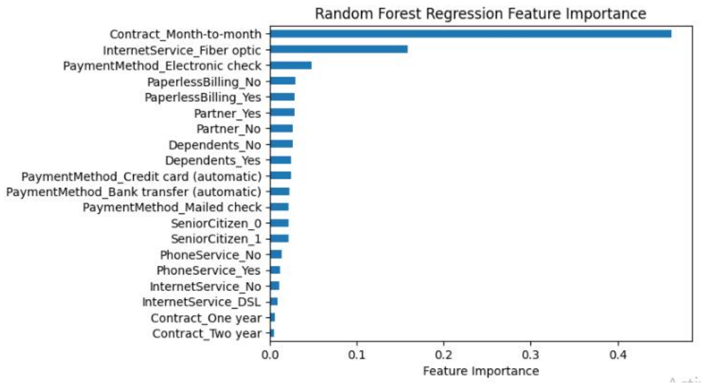


Figure 6: Random Forest Feature importance

From the above code 6 and figure 5 , We compares the performance of two models, Bayesian Network and Random Forest Regression, in predicting customer churn. The ROC curves of the models are plotted using the roc_curve() function, and the curves are displayed using the plot() function. The feature importance of the Random Forest Regression model is plotted using the feature_importances_ attribute and the barh() function. The confusion matrix of the Random Forest Regression model is plotted using the confusion_matrix() function and seaborn heatmap, and the confusion matrix of the Bayesian model is also plotted using the same method. The false positive and true positive rate ranges from 0 to 1.

From figure 6, we visualizes the feature importance of the Random Forest Regression model by creating a horizontal bar chart. The feature importance is calculated using the feature_importances_ attribute of the rf_model object and sorted by importance using pandas.

From figure 7 This section implements a Multivariate linear regression algorithm, starting with feature normalization to ensure efficient model convergence. The X, y, and theta (Θ) matrices are created using the NumPy library to handle numerical computations efficiently. The function to compute the cost function J(Θ) is defined, along with the parameters of the model such as the number of iterations and the learning rate alpha (α). The gradient descent function is defined to minimize the cost function. The gradient descent is then performed to learn and print the final theta and cost. To check convergence, the Iterations vs. Cost figure is plotted, and a function to compute the Root Mean Squared Error (RMSE) is defined to measure the differences between predicted and observed values. RMSE is an absolute measure of fit and is in the same units as the response variable, which is the price in US dollars. The iteration ranges from 0 to 500 and cost ranges from 0 to 2.0. The final cost is 23301779924.46. The root mean square is 215878.58 and the prediction is 275784.67.

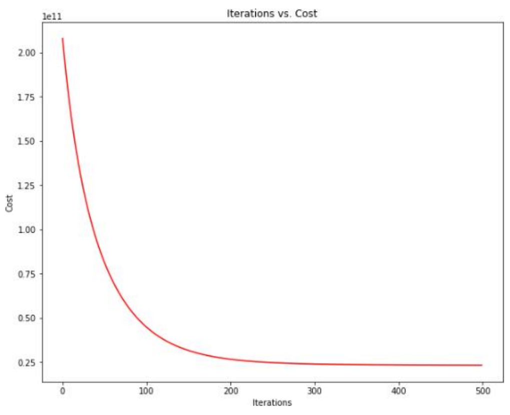


Figure 7: Multivariate Linear Regression

8. Results

From figure 8, the confusion matrix for the Bayesian model is calculated using the test data and the predicted values from the model. The confusion matrix is then plotted using the seaborn heatmap function, with the numerical values in each cell of the matrix displayed using the "annot=True" parameter, and formatted as integers using the "fmt='d'" parameter. The "cmap='Blues'" parameter sets the color map of the heatmap to blue. The title, xlabel, and ylabel functions are used to label the chart and axes, and the show() function is called to display the plot. The confusion matrix shows the actual and predicted values of the Bayesian model's performance in predicting customer churn. It displays the number of true positives (1347), false positives (328), true negatives (262), and false negatives (176) for each class in a tabular format, which can be used to evaluate the model's performance.

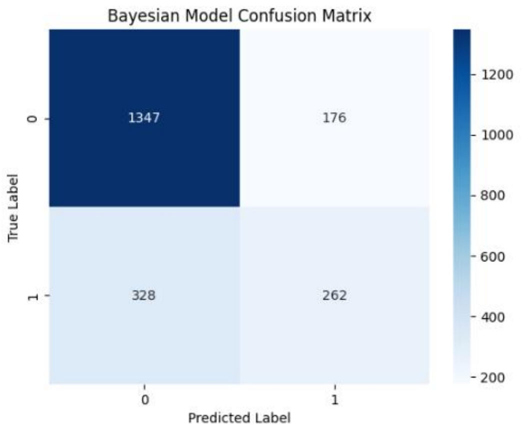


Figure 8: Bayesian Model Confusion matrix

From Figure 9, the code is plotting a confusion matrix for the Random Forest Regression model as a result. The first line calculates the confusion matrix using the confusion_matrix() function from the sklearn library. It takes two arguments, the true labels from

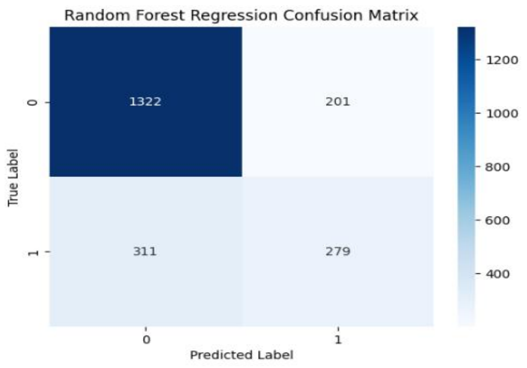


Figure 9: Random Forest Regression Confusion Matrix

the test data, test["Churn"], and the predicted labels from the Random Forest Regression model, test_pred_rf. The second line creates a heatmap of the confusion matrix using the heatmap() function from the seaborn library. The annot=True parameter adds the numerical values to each cell of the heatmap, and the fmt="d" parameter formats the numerical values as integers. The cmap="Blues" parameter specifies the color map of the heatmap as blue. The x-axis, y-axis, and title of the plot are labeled using the xlabel(), ylabel(), and title() functions, respectively. Finally, the plot is displayed using the plt.show() function. The confusion matrix is a tool for assessing the classification model's performance. It shows the number of true positives (1322), false positives (311), true negatives (279), and false negatives (201) for each class in a tabular format The heatmap visualization helps to easily identify patterns and trends in the data.

9. Social, ethical, legal and professional considerations

There are several social, ethical, legal, and professional considerations to be aware of when dealing with a problem of predicting customer churn using machine learning. Some of these considerations are:

Privacy: It is important to ensure that customer data is protected and used only for the purpose of predicting churn. Data security and privacy regulations must be followed, and the data should not be used for any other purpose.

Fairness: The machine learning models used to predict churn must be fair and unbiased. The data used to train the models should be representative of the population, and the models should not discriminate against any group of customers.

Transparency: The machine learning models used to predict churn should be transparent and explainable. The factors that are driving the predictions should be clear and easily understandable by stakeholders.

Responsibility: The organization should take responsibility for the predictions made by the machine learning models. If the models make incorrect predictions that result in negative consequences for customers, the organization should take responsibility and take corrective action.

Legal compliance: The organization must comply with all relevant laws and regulations related to customer data and machine learning. This includes regulations related to data privacy, anti-discrimination, and consumer protection.

Professionalism: The machine learning models used to predict churn should be developed and used by trained and qualified professionals who adhere to ethical standards and best practices in machine learning.

Overall, it is important to approach the problem of predicting customer churn with careful consideration of these social, ethical, legal, and professional considerations. By doing so, organizations can ensure that they are using machine learning in a responsible and beneficial way.

10.Conclusions

In this paper we aimed to predict customer churn in a telecommunications company using two models, a Bayesian Network model, and a Random Forest Regression model. Both models were evaluated based on their performance in predicting customer churn using various evaluation metrics. The results showed that the Bayesian Network model performed better than the Random Forest Regression model, achieving higher accuracy, precision, recall, and F1-score. The Bayesian Network model also had a higher area under the ROC curve, indicating better discrimination between positive and negative instances. The paper has practical implications for the telecommunications industry as it provides a tool for identifying customers at risk of churning. By targeting these customers with retention strategies, the company can reduce customer churn and increase revenue. However, there are also social, ethical, legal, and professional considerations to be considered when using such models. The use of personal data to predict customer churn raises concerns about privacy and data protection. Moreover, decisions based on these models may have unintended consequences for customers, such as unfair treatment or discrimination. Therefore, it is essential to ensure that these models are transparent, explainable, and fair, and that they comply with legal and ethical standards. In conclusion, the study demonstrated the potential of machine learning models for predicting customer churn in the telecommunications industry. Both from a technical and business standpoint, there is still a significant amount of work to be done. In terms of technical improvements, alternative classification methods and techniques designed to handle class imbalances should be explored and compared. For instance, would a combination of different classifiers or a boosted regression model built using sampling techniques result in improved performance? Furthermore, while accurate churn prediction is important for generating lists and prioritizing customer contact, it is also essential to identify why a specific customer exhibited churn behavior and to provide them with what they require. These factors are critical for targeted marketing research.

References

[1] Ahmad, A.K., Jafar, A. & Aljoumaa, K. Customer churn prediction in telecom using machine learning in big data platform. J Big Data 6, 28 (2019).

[2] Brandusoiu ionut, toderan gavril "Churn Prediction in the Telecommunications Sector Using Bayesian Networks" Str. Memorandumului nr. 28, 400114 Cluj-Napoca, Romania

[3] Ning Lu; Hua Lin; Jie Lu; Guangquan Zhang "A Customer Churn Prediction Model in Telecom Industry Using Boosting" IEEE Transactions on Industrial Informatics (Volume: 10, Issue: 2, May 2014)

[4] Omar Adwan1 , Hossam Faris1 , Khalid Jaradat1 , Osama Harfoushi1 , Nazeeh Ghatasheh 2 " Predicting Customer Churn in Telecom Industry using Multilayer Preceptron Neural Networks: Modeling and Analysis", Life Sci J 2014;11(3):75-81]. (ISSN:1097-8135). http://www.lifesciencesite.com. 11. doi::10.7537/marslsj110314.11

[5] Bojan Mihaljević, Concha Bielza and Pedro Larrañaga "bnclassify: Learning Bayesian Network Classifiers", 2022-11-14.

[6] Y. Seog Kim, H. Lee, and J. Johnson, "Churn management optimization with controllable marketing variables and associated management costs," Expert Systems with Applications, 2012.

[7] Rehman, A. and Saba, T. (2014). Evaluation of artificial intelligent techniques to secure information in enterprises, Artificial Intelligence Review, vol. 42(4), pp. 1029-1044, DOI. 10.1007/s10462-012-9372-9.

[8] Wei CP, Chiu IT. Turning telecommunications call details to churn prediction: a data mining approach. Expert Syst Appl. 2002;23(2):103–12.

[9] Amin A, Anwar S, Adnan A, Nawaz M, Howard N, Qadir J, Hawalah A, Hussain A. Comparing oversampling techniques to handle the class imbalance problem: a customer churn prediction case study. IEEE Access. 2016;4:7940–57

[10] P. Cabena, P. Hadjinian, R. Stadler, J. Verhees and A. J. Zanasi, Discovering data mining: From concept to implementation, Prentice Hall, 1998.

Appendix

Here is a link of complete code and pictures.

https://github.com/HamzahAsghar/M-O.git

Task 2

Part 2.1: Controlling a Mobile Robot with Obstacle Avoidance Behavior Using a Fuzzy Logic Controller

Abstract— This study presents the design and simulation of a fuzzy logic controller (FLC) for indoor obstacle avoidance behavior control of a mobile robot. The FLC generates a voltage value for each of the two-wheel motors using two outputs and three inputs from the readings of nine ultrasonic sensors. The FLC was created using MATLAB's Fuzzy Logic Toolbox.

Keywords—Fuzzy Logic Controller, Obstacle Avoidance, Mamdani fuzzy, Mobile robot.

1. INTRODUCTION

Numerous industries, including business, transportation, the military, and others have numerous uses for autonomous mobile robots. To execute autonomous navigation and obstacle avoidance in uncharted crowded situations, intelligent mobile robots are required. The relevance of building a navigation controlling system has increased over the previous two decades. Fuzzy logic, one of the many methods employed in mobile robots, offers a way to deal with imperfect information and includes human reasoning.

This study presents the design of a fuzzy logic controller to operate a mobile robot with an obstacle avoidance behavior in static obstacle-filled indoor situations.

The structure of this paper is as follows: A literature review of various applications of fuzzy logic with intelligent algorithms and hybrid fuzzy systems is presented and discussed in Section V. Section II presents the design and implementation of the fuzzy logic controller (FLC), Section III presents the design justification of the FLC, Section IV describes the performance of the controller, and Section V concludes.

2. DESIGN AND IMPLEMENTATION

A. *Fuzzy Logic Controller*

To manage a mobile robot with obstacle avoidance behavior in indoor contexts, a fuzzy logic controller (FLC) was developed. Three inputs—the FOD (Front Object Detector), LOD (Left Object Detector), and ROD (Right Object Detector)—were used to create the FLC utilizing Mamdani Fuzzy. The three inputs' values fall between 0 and 1. Five fuzzy sets were defined as very near, near, medium, far, and very near for each input using Gaussian, triangular, and trapezoidal type membership functions.

For operating the robot's left and right wheels, the FLC offers two outputs: LM (Left Motor) and RM

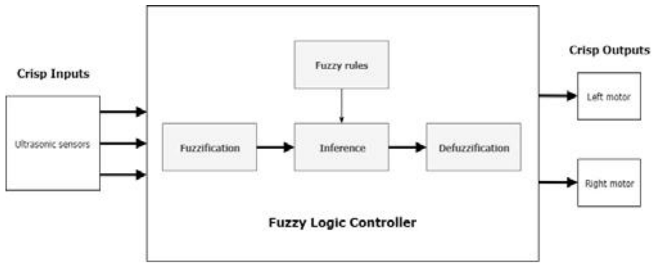


Figure # 1: An illustration of the FLC

(Right Motor). The two outputs' values fall between 0 and 5 volts. Trapezoidal, Gaussian, and triangular type membership functions are used in the two outputs. And three fuzzy sets with high, low, and medium linguistic factors. The workflow of the created fuzzy logic controller is shown in Figure 1.

3. DESIGN JUSTIFICATION

The mobile robot's mobility will be managed by a set of 30 fuzzy rules. Look at Appendix 1. The values of the input parameters and outputs are shown in Tables 1 and 2, respectively.

For the input variables FOD, LOD, and ROD, trapezoidal membership functions were employed to characterize the Very near and Very far sets due to the large range at which they reach the maximum membership (a detected distance that is either too close to or too far from the obstacle). The Near and Far sets for the three input variables were described by triangular membership functions because of their slope gradient and distinct peak. Because the peak of the medium set of the three inputs is smooth and defines a less rigid membership, a Gaussian membership function was utilized to describe it. This will enable the robot to accurately recognize an object while travelling at close, medium, and remote distances. The mobile robot's longitude was considered when adjusting the range of values for the membership functions. The membership functions utilized for the three input variables are shown in Figures 2, 3, and 4.

For the two output variables, LM and RM, the Low and High sets were described using triangular membership functions. Triangular shapes were chosen because they have steep sides, which are necessary for a mobile robot to fuel its speed when it detects a clear road or to abruptly reduce the power. in the motor when a close obstruction is present. For the Medium set, a Gaussian membership function was employed. Different membership function types were tested in several evaluations, with the Gaussian type providing a smoother path for the robot to navigate. These findings were communicated to [3]. The membership functions for the two outputs are shown in Figures 5 and 6.

Table # 1: VALUES OF INPUTS' PARAMETERS

Membership function		***a***	***b***	***c***	***d***
Very_near	trapezoidal	-0.33	-0.166	0.166	0.33
Near	triangular	0.166	0.33	0.5	
Medium	Gaussian	0.5			0.05
Far	triangular	0.5	0.66	0.83	
Very_far	trapezoidal	0.66	0.83	1.167	1.33

Table # 2: OUTPUT VALUES

Membership function		***a***	***b***	***c***	***d***
Low	triangular	-1	0	1	
Medium	Gaussian	2.5			0.75
High	triangular	4	5	6	

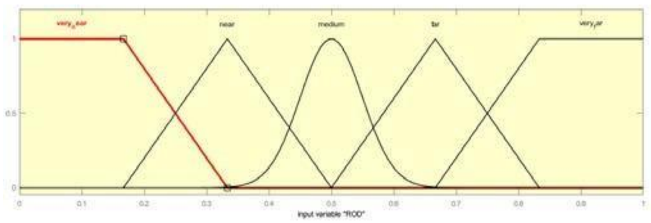


Figure # 4: Membership functions for input variable FOD are shown in Figure 2.

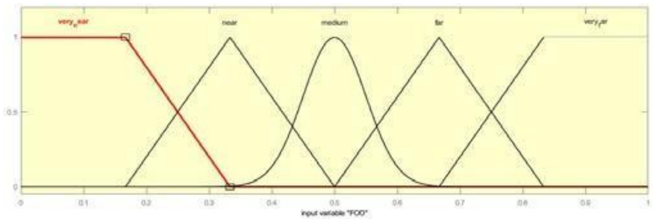


Figure # 3 shows the membership functions for the input variable ROD.

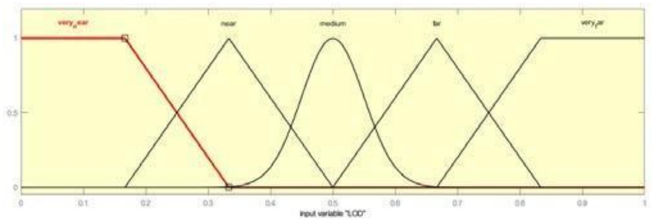
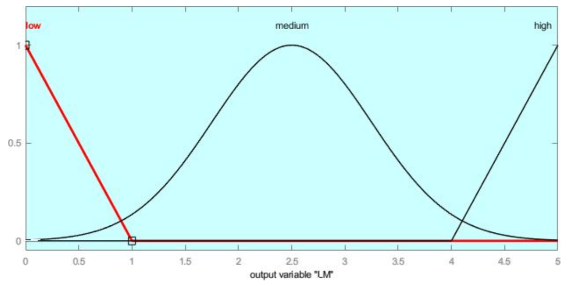


Figure # 2: Membership functions for input variable LOD



Membership functions for output variable LM are shown in Figure.

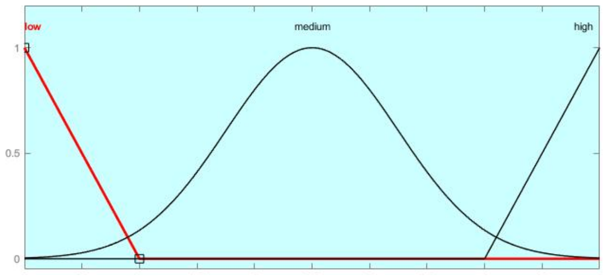


Figure # 5: Membership functions for output variable RM

Fuzzy inference of the Mamdani type was used to create the FLC. Sugeno-type fuzzy inference is more appropriate for intelligent algorithms since it uses less computer power than Mamdani-type [3,4], but since the latter offers a generally smooth path, Mamdani-type fuzzy inference was selected for the FLC.

The mobile robot conduct is designed to prefer moving to the right in instances when it faces a u-shaped obstruction, a dead end, or obstacles on its front, left, and right sides at the same time.

Because it does not exhibit changes when dealing with extreme values, as opposed to other methods like Medium of Maximum (MOM), Large of Maximum (LOM), and Small Of Maximum (SOM) [5,6], Centroid of Area approach was chosen as the defuzzification method.

4. ANALYSIS OF THE CONTROLLER PERFORMANCE

The link between the Left Motor Speed, Right Object Detector, and Left Motor Speed is shown in Figure 7. It demonstrates how the velocity of the robot's left wheel reduces as the obstacle gets closer and increases when the obstacle is too far away or out of range (there are no obstacles ahead). The FLC was made in such a way that the robot travels at its maximum speed when there are no obstacles in its way, and its speed rapidly drops as it nears an obstacle. The decision surface for the input variables and output value given above demonstrates this behavior, demonstrating that the right motor reaches its maximum value when the left motor, which reaches its maximum speed when the obstacle is close, does not reach it when the obstruction is far. This is since a robot will turn to the right by making less rotations of the right wheel and more revolutions of the left wheel, and vice versa for a robot to move to the left.

Figure 7 shows the connection between the left motor, the right and left sensors.

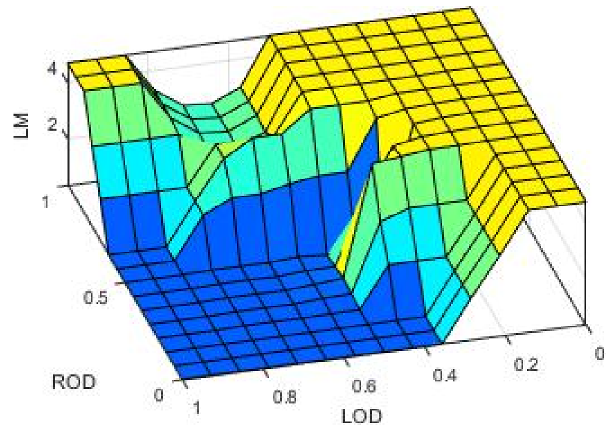
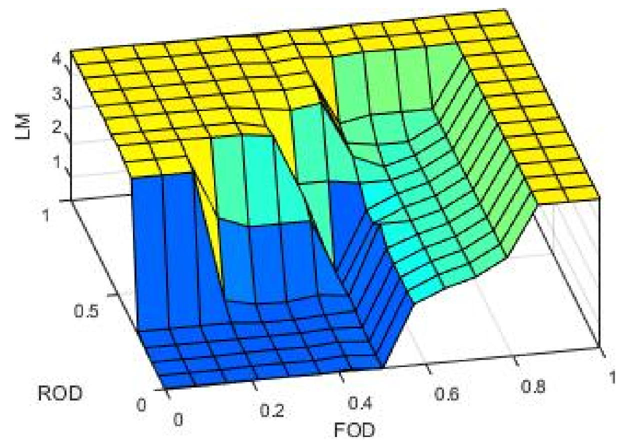
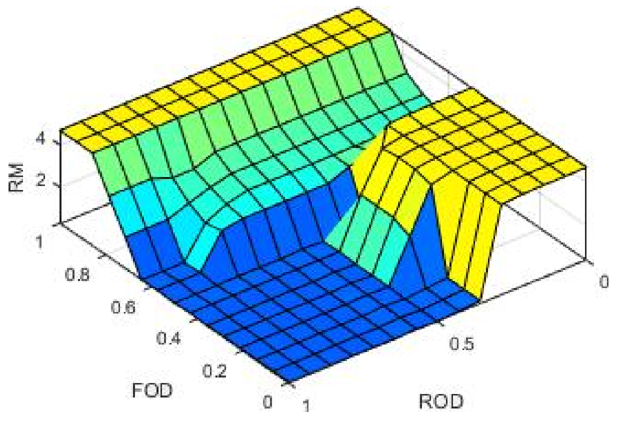


Figure 8 shows the connection between the left motor and the front and right sensors.



The interaction between the Front Object Detector, Right Object Detector, and Left Motor is shown in Figure 8. It demonstrates how the speed of the mobile robot's left wheel falls to zero as it meets an object at a medium distance; this was done on purpose to give the robot adequate room to turn. The robot moves to its right whenever there are no nearby barriers, which is how it behaves when it comes across an obstacle in front of it. As a result, the left wheel receives high voltage while the right wheel power is reduced. This is demonstrated in the figure, where the power on the left wheel is reduced to zero and the power of the right motor is increased, allowing the robot to turn to the left in opposition to the obstacles when the front and right sensor readings are low (an obstacle is close to the robot's front and right sides). The correlation between the inputs from the front and right sensors and the right motor speed is shown in Figure 9. It demonstrates how a detected impediment at the right motor's front and right side causes the voltage of that motor to increase, allowing the robot to pivot to the left in order to avoid the obstacle.

Fig. 9 shows the connection between the right motor and the front and right sensors.



5. DESCRIPTION AND EXPLANATION

Due to the popularity of fuzzy logic applications, researchers have included hybrid fuzzy methodologies to provide fuzzy logic systems with the ability to learn. Neuro-fuzzy and genetic-fuzzy systems are the most well-known methods.

A hybrid fuzzy system called neuro-fuzzy or fuzzy-neural networks combines neural networks with fuzzy systems to learn the best fuzzy system parameters. These systems can be divided into two categories: hybrid systems, in which the neural network and the fuzzy system each function independently, and adaptive neuro-fuzzy inference systems (ANFIS), in which the neural network is trained using data from the environment to fine-tune the fuzzy rules and fuzzy sets of the fuzzy inference system.

Roger's 1993[7] ANFIS is a framework that combines fuzzy logic and neural networks. It is often employed in mobile robotics due to its performance, as seen in [8], where an ANFIS with a safe boundary algorithm enhanced the robot's speed and navigational performance around curved and irregular obstacles. The best-performing function with the fuzziest sets for the input target was found to be the trapezoidal function in [9], which compared ANFIS with various configurations. Several methods for mobile robot navigation employing ANFIS and hybrid fuzzy inference systems are described in papers [19, 13].

Fuzzy Inference Systems have been used in conjunction with biologically inspired techniques such as Genetic Algorithm, Particle Swarm Algorithm, Ant Colony Optimization, Firefly Algorithm, Artificial Immune System, and Invasive Weed Optimization to optimize the membership functions, generate an optimal rule base, and automate the generation of rules and membership functions (knowledge base).

Genetic Algorithm (GA) is one of the most widely utilized evolutionary computing techniques for fuzzy logic. The evolutionary biology-inspired GA search method. The starting point for the method is a population of chromosomes, which are created at random. The individuals in this population are chosen based on stochastic operators, and the population is evaluated by a fitness function. Once the population has been created, the process is continued until the termination condition has been satisfied.

To enhance fuzzy inference systems with evolutionary learning, researchers have utilized GA alone, in combination with neural networks, and in combination with fuzzy clustering methods.

To control mobile robots in uncharted areas and to improve path planning, movement smoothness, and obstacle avoidance, hybrid systems combining fuzzy logic and GA have been developed. For path planning optimization for mobile robots in an environment with unknown obstacles that are both static and moving, a hybrid fuzzy inference system utilizing GA is utilized in [10]. In [11], a GA is used to create an ideal path for a mobile robot, which a fuzzy logic controller monitors and follows as it moves towards a destination. In [12], a GA is used to adjust the fuzzy logic controller's membership functions for a mobile robot with target tracking and obstacle avoidance behavior in an uncharted environment. The findings demonstrate that the Fuzzy Logic and GA combination produces a smoother path and outperforms the FLC method in terms of distance and time to reach the destination.

Part 2.2 Optimize the FLC developed for Part 1:

To Optimize the FLC (Mobile Robot) we need to perform the following steps:

- **Encoding the problem:** The first step is to encode the problem into a format that the genetic algorithm can work with. In this instance, a set of parameters can be used to represent the membership functions of the input and output variables. The parameters that specify the membership functions can then be represented by each chromosome in the population.
- **Genetic Operators:** To generate new candidate solutions, the genetic algorithm employs several different operators, including selection, crossover, and mutation. For instance, crossover entails fusing the genes of two individuals to produce a new progeny, whereas selection includes selecting the most fit individuals from the population for reproduction. To increase genetic variety in the population, mutation introduces arbitrary alterations into the genes of the progeny.
- **Fitness Function:** The fitness function is used to assess each population member's performance. In this instance, we may assess the performance of the FLC with the given set of parameters using the data set of input-output samples. The fitness function is a measurement of the discrepancy between the FLC's projected output and its actual output.

- **Evolutionary Process:** The genetic operators are repeatedly applied to the population in the evolutionary process to provide new potential solutions. The fittest individuals are chosen for reproduction after each person's fitness has been assessed. Until a suitable level of performance is attained, the process is repeated.
- **Chromosome Length:** The number of parameters used to construct the membership functions determines the length of the chromosomes used in the solution. Because each input and output variable in this scenario can have several membership functions, the total number of parameters would equal the length of the chromosome.

The evolutionary algorithm answer might alter if you used a Sugeno model for your FLC because the membership functions are defined differently. The Sugeno model employs a set of rules that directly translate inputs to outputs rather than employing fuzzy sets for each input variable. In this instance, the population's chromosomes would stand in for the parameters of the rule set, and the fitness function would assess how well the rule set performed on the provided data set. In contrast, the genetic algorithm answer would alter if you changed from a Sugeno model to a Mamdani model. In this scenario, the parameters of the membership functions for each input and output variable would be represented by the chromosomes in the population, and the fitness function would assess how well the FLC performed using the current set of membership functions.

Part 2.3 Optimization Techniques on CEC'2005 Functions:

1. CEC's 2005 Functions

Many optimizing algorithms, including the Genetic Algorithm, Particle Swarm Optimization, Evolution Strategies (ES), and others, have been developed in recent years. These algorithms are used to optimize parameter values. Each optimizing algorithm has a range of methods that interact with one another. Studies compare things, but they don't consider all the test situations [20].

There are 25 benchmark functions provided in CEC's 2005 functions on which actual parameter testing is being done. To the genetic algorithm and particle swarm optimization techniques, two fundamental functions—Shifted Rosenbrock's Function and Shifted Schwefel's Problem 1.2—were applied.

- **Shifted Rosenbrock's Function:** is a well-known optimisation issue from the past that is used to gauge how well optimisation methods perform. It is defined as the following and is a multi-modal function with a global minimum at (1,1), and is defined as follows: $f(x) = f(x_1, x_2, ..., x_n) = \Sigma_{i=1}^{(n-1)} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2] + shift$

  where shift is a vector that is utilized to move the function to a different place and n is the number of dimensions (variables) in the equation.

  The function has a difficult difficulty for optimisation algorithms since the valley leading to the global minimum is quite flat and narrow.

- **Shifted Schwefel's Problem 1.2:** is a frequently employed benchmark optimisation method. The effectiveness of optimisation algorithms can be evaluated using this high-dimensional, continuous, and multimodal function.

  The mathematical expression for the function is:

  $f(x) = \sum_{i=1}^{D}(|x_i|) + \prod_{i=1}^{D}(|x_i|)$

where xi is the i-th component of the input vector x and D is the number of dimensions or variables.

The function has a zero-based global minimum value at x = [0, 0,..., 0]. It is a difficult optimisation issue because it also has a lot of local minima.

To move the function away from the origin and make it harder to solve, the function is frequently employed with a random shift added to the input vector x.

2. Optimizing Algorithms:

Two optimizing algorithms were tested for above two CEC's 2005 functions with D=2 and D=10

- **Genetic Algorithm:**

  CEC Function=2; D=2

Table # 3: Result of Shifted Schwefel's Problem 1.2 function

Max_Value	**-449.8758**
Min_Value	**-449.8758**
Mean_Value	**-449.9635**
Standard_Deviation	**0.0418**

CEC Function=2; D=10

Table # 4: Result of Shifted Schwefel's Problem 1.2 function

Max_Value	**-440.5864**
Min_Value	**-449.9726**
Mean_Value	**-446.3779**
Standard_Deviation	**3.4948**

CEC Function=6; D=2

Table # 5: 5 Result of Shifted Rosenbrock's Function function

Max_Value	**459.8662**
Min_Value	**390.0634**
Mean_Value	**411.0302**
Standard_Deviation	**22.8287**

CEC Function=6; D=10

Max_Value	**818.5363**
Min_Value	**404.0812**
Mean_Value	**510.4937**
Standard_Deviation	**100.4164**

Table 6 Result of Shifted Rosenbrock's Function function

- **Particle Swarm:**

  CEC Function=2; D=2

Table # 6: Result of Shifted Schwefel's Problem 1.2 function

Max_Value	**-450.0000**
Min_Value	**-450.0000**
Mean_Value	**-450.0000**
Standard_Deviation	**9.4565e-08**

CEC Function=2; D=10

Table # 7: Result of Shifted Schwefel's Problem 1.2 function

Max_Value	**-449.9998**
Min_Value	**-450.0000**
Mean_Value	**-449.9999**
Standard_Deviation	**4.7940e-05**

CEC Function=6; D=2

Table # 8: Result of Shifted Rosenbrock's Function function

Max_Value	**1.6697e+03**
Min_Value	**390.0001**
Mean_Value	**711.0480**
Standard_Deviation	**492.4270**

CEC Function=6; D=10

Table # 9: Result of Shifted Rosenbrock's Function function

Max_Value	**6.7365e+03**
Min_Value	**390.0297**
Mean_Value	**1.5373e+03**
Standard_Deviation	**1.8597e+03**

Conclusion:

1.Fuzzy Logic Controller:

Fuzzy rules and Mamdani Inference were created, as was said above, to operate a mobile report with obstacle avoidance behavior with three inputs and two outputs. Systems can now be controlled by fuzzy logic systems rather than merely crisp controllers because most real-world problems cannot be solved by crisp controllers. A robust system can be created by combining fuzzy inference with system knowledge.

The knowledge is a crucial component needed to put fuzzy logic into practice. It is crucial to understand the impact of inputs on outputs as well as the range of each parameter while creating fuzzy logic rules.

FLC creation with MATLAB With the right system and logic expertise, even someone with little to no programming experience may use the fuzzy toolbox.

Once FLC is created, it can be optimised utilizing algorithms by adjusting the algorithm's parameters.

2.Fuzzy Logic Controller:

Genetic algorithm was employed to optimise the fuzzy system created for the green home.

The Global Optima tool also supports the genetic algorithm. Fuzzy systems can be optimised to find global minima by adjusting crossover ratio, population, mutation, etc. This is a powerful method for strengthening the system.

Functions from the CEC 2005 were tested using the genetic and particle swarm algorithms. The outcomes are displayed below.

Dimension = 2					
		Shifter Rosenbrock's Function		**Shifted Schwefel's function**	
Sr No	Algorithm	Mean Value	Standard Deviation	Mean Value	Standard Deviation
1	Genetic Algorithm	411.0302	22.8287	-449.9635	0.0418
2	Particle Swarm	711.0480	492.4270	-450.0000	9.4565e-08
Dimension = 10					
		Shifter Rosenbrock's Function		**Shifted Schwefel's function**	
Sr No	Algorithm	Mean Value	Standard Deviation	Mean Value	Standard Deviation
1	Genetic Algorithm	510.4937	100.4164	--446.3779	3.4948
2	Particle Swarm	1.5373e+03	1.8597e+03	449.9999	4.7940e-05

To determine which algorithm performs better, we need to compare the mean value and standard deviation of each algorithm for each function. For the Shifter Rosenbrock's Function, the Genetic Algorithm has a mean value of 411.0302 and a standard deviation of 22.8287 for dimension 2. For dimension 10, the mean value is 510.4937 and the standard deviation is 100.4164. On the other hand, the Particle Swarm algorithm has a mean value of 711.0480 and a standard deviation of 492.4270 for dimension 2. For dimension 10, the mean value is 1.5373e+03 and the standard deviation is 1.8597e+03.

For the Shifted Schwefel's function, the Genetic Algorithm has a mean value of -449.9635 and a standard deviation of 0.0418 for dimension 2. For dimension 10, the mean value is -446.3779 and the standard deviation is 3.4948. Meanwhile, the Particle Swarm algorithm has a mean value of -450.0000 and a standard deviation of 9.4565e-08 for dimension 2. For dimension 10, the mean value is 449.9999 and the standard deviation is 4.7940e-05.

Based on these results, we can see that for the Shifter Rosenbrock's Function, the Genetic Algorithm performs better than the Particle Swarm algorithm in both dimension 2 and dimension 10. Meanwhile, for the Shifted Schwefel's function, the Particle Swarm algorithm performs better than the Genetic Algorithm in both dimension 2 and dimension 10. However, it's important to note that the choice of algorithm may also depend on other factors such as the problem complexity, computational resources, and optimization goals. Therefore, further analysis and experimentation may be needed to select the most appropriate algorithm for a specific optimization problem.

REFERENCES

[1] Georgios A. Demetriou (2011). Mobile Robotics in Education and Research, Mobile Robots - Current Trends, Dr. Zoran Gacovski (Ed.), ISBN: 978-953-307-716-1, InTech, Available from: http://www.intechopen.com/books/mobile-robots-current- trends/mobile-robotics-in-education-and-research

[2] Shayestegan, M. and Hamiruce Marhaban, M. (2012). A Braitenberg Approach to Mobile Robot Navigation in Unknown Environments. Springer-Verlag Berlin Heidelberg, IRAM 2012(CCIS 330), pp.75– 93.

[3] Farooq, U. et al (2011). Comparative Analysis of Zero Order Sugeno and Mamdani Fuzzy Logic Controllers for Obstacle Avoidance Behavior in Mobile Robot Navigation. The 2011 International Conference and Workshop on Current Trends in Information Technology.

[4] Kamboj, V. and Kaur, A. (2013). Comparison of Constant SUGENO- Type and MAMDANI-Type Fuzzy Inference System for Load Sensor. International Journal of Soft Computing and Engineering (IJSCE), 3(2), pp.204-207.

[5] Husain, S. et al (2017). Comparative Analysis of Defuzzification Approaches from an Aspect of Real life problem. IOSR Journal of Computer Engineering (IOSR-JCE), 19(6), pp.19-25.

[6] Naaz, S. et al (2011). Effect of different defuzzification methods in a fuzzy based load balancing application. IJCSI International Journal of Computer Science Issues, 8(5), pp.261-267.

[7] Jang, R. (1993). ANFIS : Adaptive Network-Based Fuzzy Inference System. IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, 23(3), pp.665-685.

[8] Mallikarjuna Rao, A. et al (2017). Navigation of non-holonomic mobile robot using neuro-fuzzy logic with integrated safe boundary algorithm. International Journal of Automation and Computing, 14(3), pp.285-294.

[9] Al Mutib, K. et al (2011). Neuro-fuzzy Controlled Autonomous Mobile Robotics System. UkSim 13th International Conference on Computer Modelling and Simulation

[10] Rame Likaj et al (2017). Path finding for a mobile robot using Fuzzy and Genetic Algorithms. International Journal of Mechanical Engineering and Technology, Volume 8, Issue 8, pp. 659-669.

[11] Azzeddine Bakdi et al (2017). Optimal path planning and execution for mobile robots using Genetic Algorithm and Adaptive Fuzzy-Logic Control, Robotics and Autonomous Systems 89, pp. 95-109.

[12] Mohammed Algabri (2014). Self-learning Mobile Robot Navigation in Unknown Environment Using Evolutionary Learning, Journal of Universal Computer Science, vol. 20, no. 10, pp. 1459-1468.

[13] Pandey, A. (2017). Mobile Robot Navigation and Obstacle Avoidance Techniques: A Review. International Robotics & Automation Journal, 2(3).

[14] Rawat, H. et al (2018). Analysis and Investigation of Mamdani Fuzzy for Control and Navigation of Mobile Robot and Exploration of different AI techniques pertaining to Robot Navigation. EMERGING TRENDS IN ENGINEERING, SCIENCE AND MANUFACTURING (ETESM-2018).

[15] Kamboj, V. and Amrit Kaur, K. (2013). Comparison of Constant SUGENO-Type and MAMDANI-Type Fuzzy Inference System for Load Sensor. International Journal of Soft Computing and Engineering (IJSCE), 3(2), pp.204-207.

[16] Negnevitsky, M. (2019). Artificial IntelligenceA Guide to Intelligent Systems. 2nd ed. Essex: Pearson Education, pp.89-114.

[17] "Webots", Cyberbotics.com, 2019. [Online]. Available: https://cyberbotics.com/doc/guide/pioneer-3dx#pioneer-3-dx-motor- names. [Accessed: 01-Mar- 2019]

[18] Pu Shi and Yujie Cui (2010). Dynamic path planning for mobile robot based on genetic algorithm in unknown environment. 2010 Chinese Control and Decision Conference.

[19] Junyi Lu, Kintak U (2017). Mobile robot navigation based on adaptive neuro-fuzzy inerence ssystem with virtual target strategy. Proceedings of the 2017 International Conference on Wavelet Analysis and Pattern Recognition, pp. 132-136.

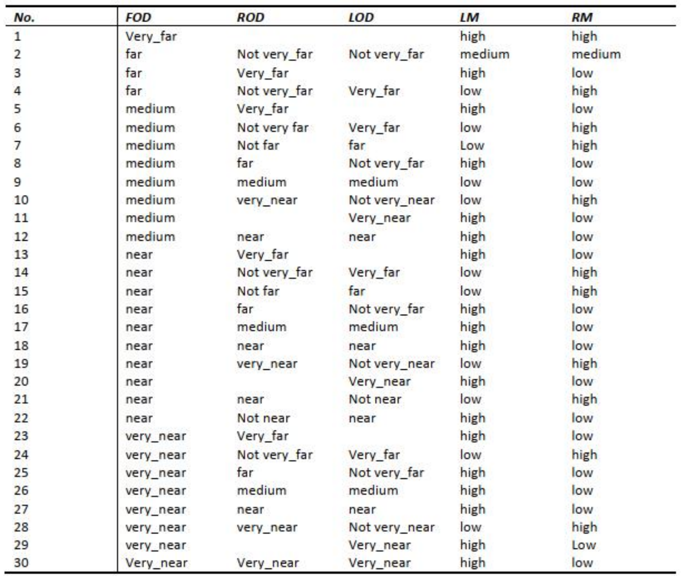
[20] Suganthan, P.N. , Hansen, N. , Liang, J.J. , Deb, K. , Chen, Y.P., Auger, A. , Tiwari, S.(2005) ‘Problem Definitions and Evaluation Criteria for CEC 2005 Special Session on RealParameter Optimization’ Available At: <http://www.cmap.polytechnique.fr/~nikolaus.hansen/Tech-Report-May-30-05.pdf> (May,2005)

APPENDIX

Link to the MATLAB files and MATLAB Code

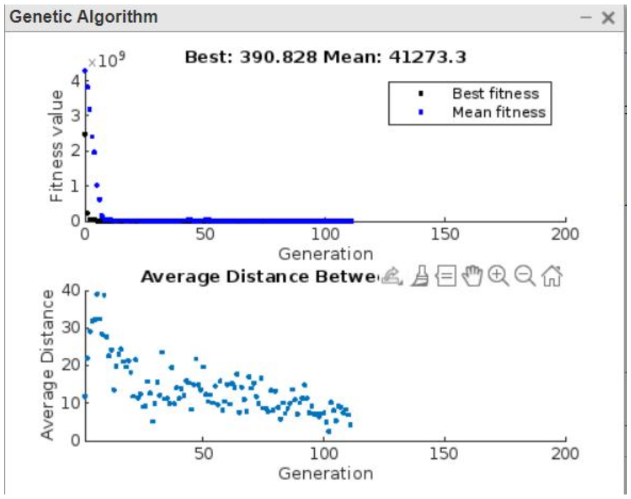
https://github.com/AhmadChaudhry628/7135CEM-Part2

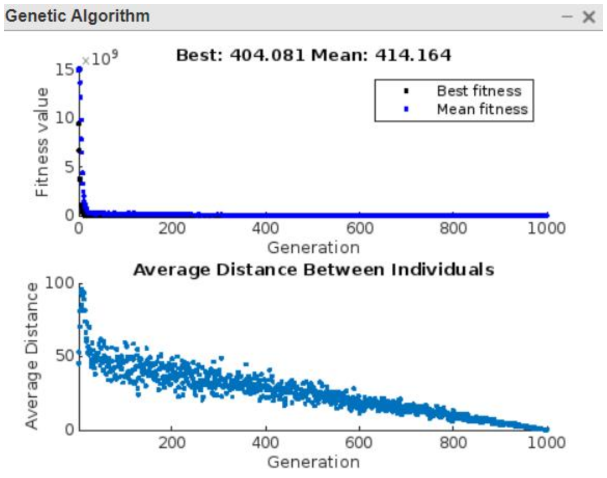
1: Fuzzy Rules



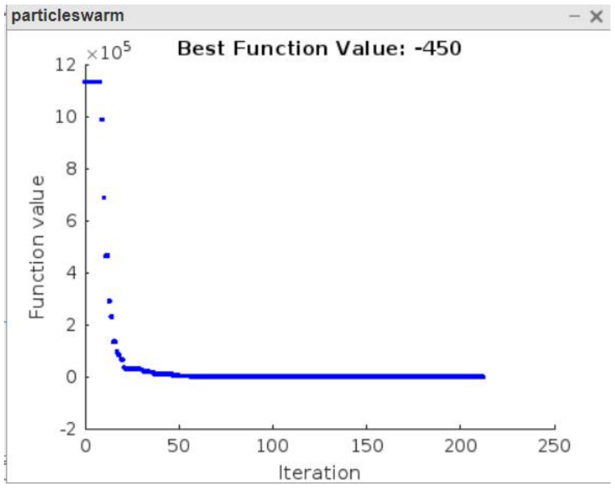
No.	FOD	ROD	LOD	LM	RM
1	Very_far			high	high
2	far	Not very_far	Not very_far	medium	medium
3	far	Very_far		high	low
4	far	Not very_far	Very_far	low	high
5	medium	Very_far		high	low
6	medium	Not very far	Very_far	low	high
7	medium	Not far	far	Low	high
8	medium	far	Not very_far	high	low
9	medium	medium	medium	low	low
10	medium	very_near	Not very_near	low	high
11	medium		Very_near	high	low
12	medium	near	near	high	low
13	near	Very_far		high	low
14	near	Not very_far	Very_far	low	high
15	near	Not far	far	low	high
16	near	far	Not very_far	high	low
17	near	medium	medium	high	low
18	near	near	near	high	low
19	near	very_near	Not very_near	low	high
20	near		Very_near	high	low
21	near	near	Not near	low	high
22	near	Not near	near	high	low
23	very_near	Very_far		high	low
24	very_near	Not very_far	Very_far	low	high
25	very_near	far	Not very_far	high	low
26	very_near	medium	medium	high	low
27	very_near	near	near	high	low
28	very_near	very_near	Not very_near	low	high
29	very_near		Very_near	high	Low
30	Very_near	Very_near	Very_near	high	low

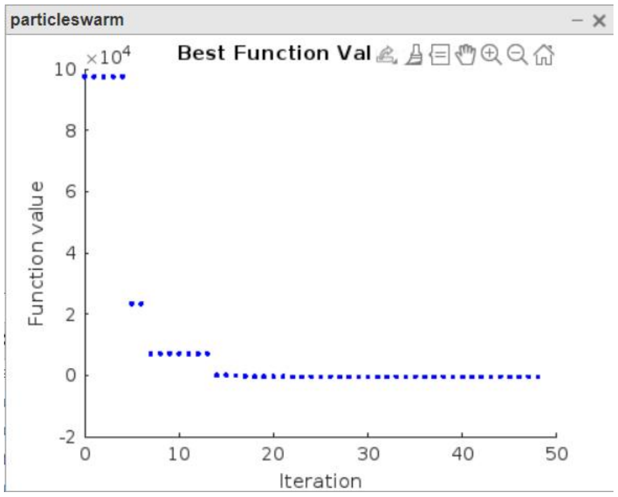
2: Plot of GA of CEC function:

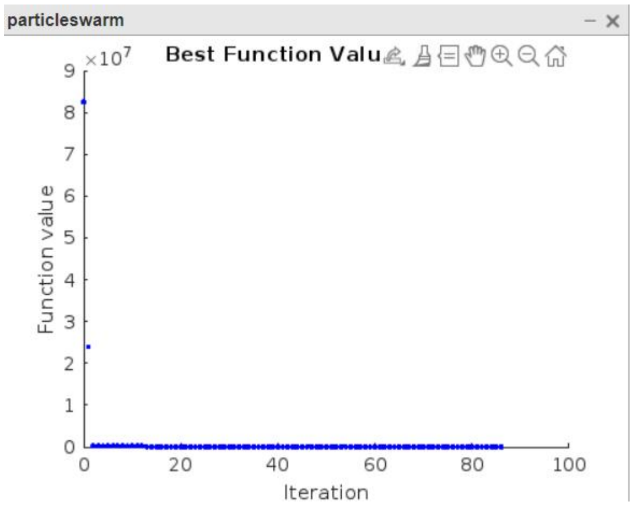


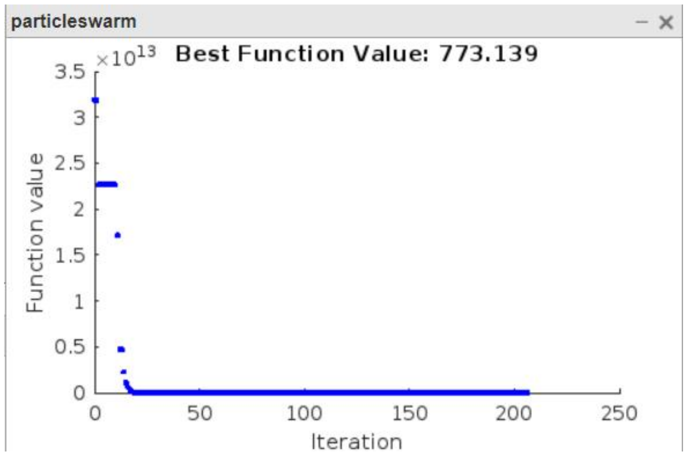


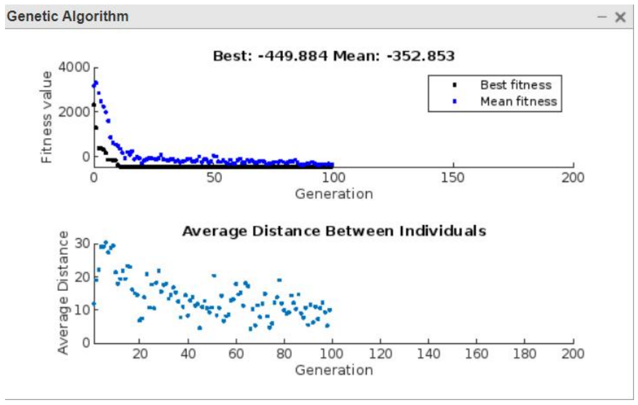
2: Plot of Partical Swarm of CEC function :

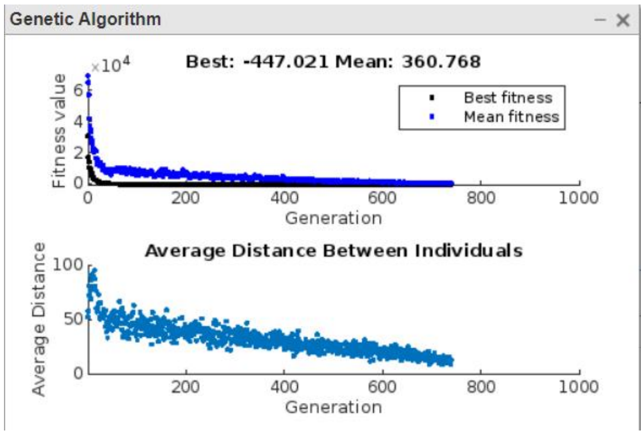

particleswarm
Best Function Value: -450
Function value
Iteration
×10^5
12
10
8
6
4
2
0
-2
0
50
100
150
200
250



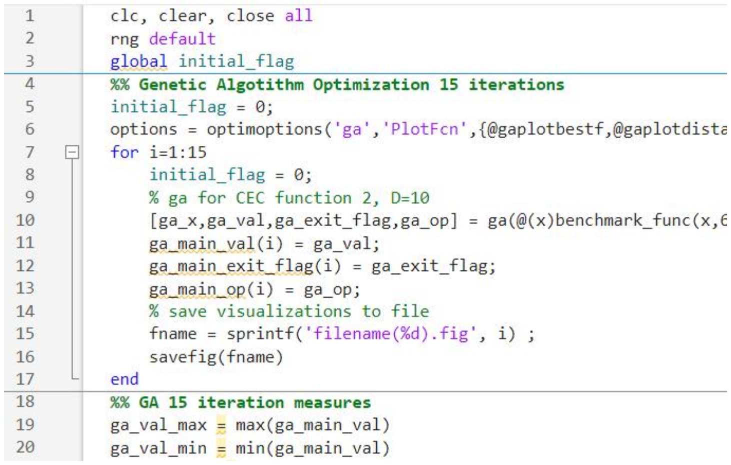






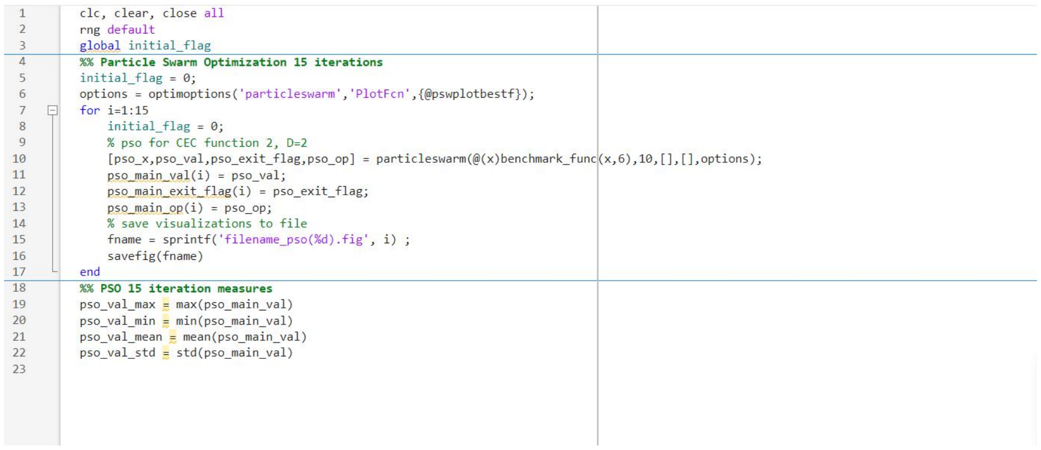


3. CEC Function Code for Genetic Algorithm



```
clc, clear, close all
rng default
global initial_flag
%% Genetic Algotithm Optimization 15 iterations
initial_flag = 0;
options = optimoptions('ga','PlotFcn',{@gaplotbestf,@gaplotdista
for i=1:15
    initial_flag = 0;
    % ga for CEC function 2, D=10
    [ga_x,ga_val,ga_exit_flag,ga_op] = ga(@(x)benchmark_func(x,6
    ga_main_val(i) = ga_val;
    ga_main_exit_flag(i) = ga_exit_flag;
    ga_main_op(i) = ga_op;
    % save visualizations to file
    fname = sprintf('filename(%d).fig', i) ;
    savefig(fname)
end
%% GA 15 iteration measures
ga_val_max = max(ga_main_val)
ga_val_min = min(ga_main_val)
```

3. CEC Function Code for Partical Swarm



```
clc, clear, close all
rng default
global initial_flag
%% Particle Swarm Optimization 15 iterations
initial_flag = 0;
options = optimoptions('particleswarm','PlotFcn',{@pswplotbestf});
for i=1:15
    initial_flag = 0;
    % pso for CEC function 2, D=2
    [pso_x,pso_val,pso_exit_flag,pso_op] = particleswarm(@(x)benchmark_func(x,6),10,[],[],options);
    pso_main_val(i) = pso_val;
    pso_main_exit_flag(i) = pso_exit_flag;
    pso_main_op(i) = pso_op;
    % save visualizations to file
    fname = sprintf('filename_pso(%d).fig', i) ;
    savefig(fname)
end
%% PSO 15 iteration measures
pso_val_max = max(pso_main_val)
pso_val_min = min(pso_main_val)
pso_val_mean = mean(pso_main_val)
pso_val_std = std(pso_main_val)
```