

MISBEHAVIOR DETECTION IN VANETS

Guided By: Dr. Dinesh Singh

TEAM DETAILS

Team Group: CS A2

Aditya Raju Darji	20194008
Deepesh Manoj Rathi	20194066
Aryan Khandelwal	20194142
Anand Tripathi	20194119
Adhitya Balaji	20194134

TOPICS TO BE COVERED

- 1 PROBLEM STATEMENT
- 2 INTRODUCTION
- 3 IDEATION
- 4 ALGORITHM
- 5 THE END

PROBLEM STATEMENT

To successfully identify the attacker from attacker's id using Machine Supervised Learning.

Use K-NN , RF and Logistic Regression and then combine the results {stack} and to filter out the attacker from normal user.

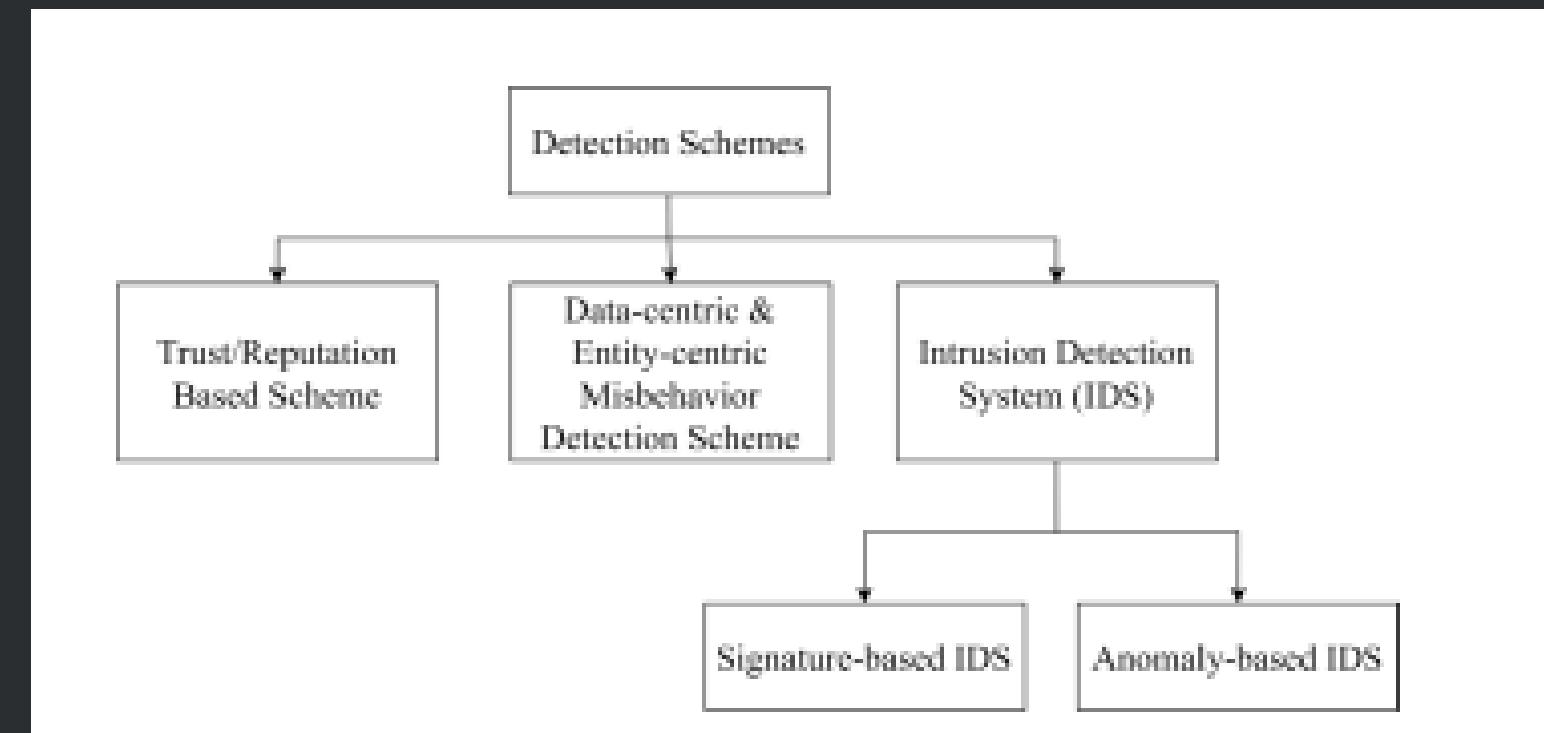


INTRODUCTION

To solve the above problem of position falsification through attacks on VANETs is checked by machine learning.

Different types of attacks, like The associate editor coordinating the review of this manuscript and approving it for publication was Tai-hoon Kim. Sybil, DoS (denial-of-service), black-hole, and false data injection attacks, may occur on the V2V or V2I communications.

These attacks can be dealt with trust/reputation based schemes, data-centric and entity-centric misbehavior detection schemes, and intrusion detection systems (IDS) as depicted

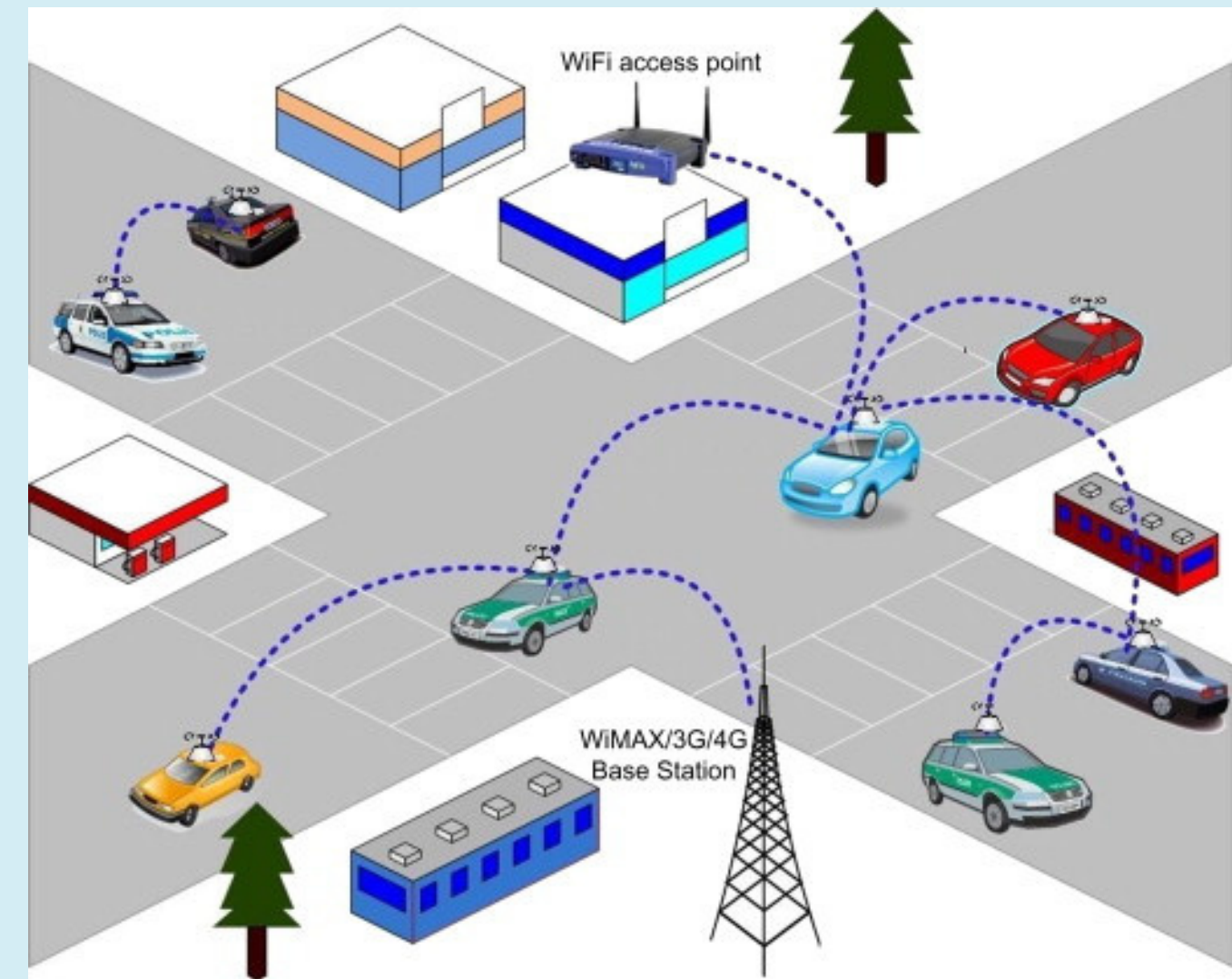


WHAT IS VANET?

Vehicular ad hoc network (VANET) is a subclass of mobile ad hoc networks (MANETs) where it is developed by moving vehicles .

VANET is getting progressively well known in rush hour gridlock administration particularly in a portion of the created nations.

It can be ordered into well-being-related application where it can spare a large number of lives every day and non-security applications for business reason.



MISBEHAVIOR DETECTION

A trusted central authority collects the data from vehicles and RSUs, and then detects the false information by using an XML dependency tree.

A centralized station which distributes certified pseudonyms to vehicles, confirms whether an attack has occurred, while checking the coarse-grained hash value of pseudonyms.

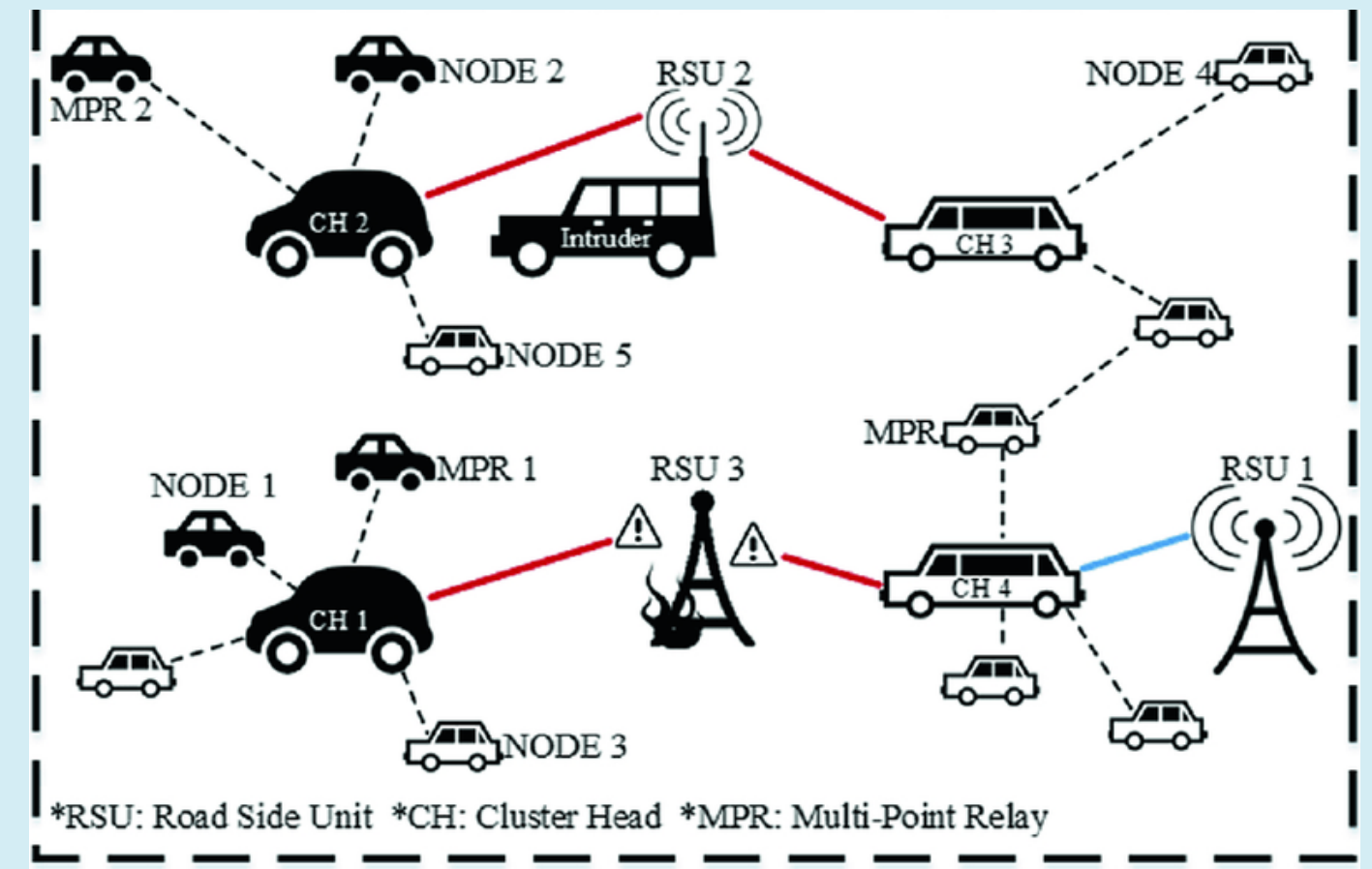
In this system, detection is done by vehicles considering the load, the distrust value (or reputation), and the distance. Then in case of an attack, vehicles inform the cluster head, which updates the distrust value and broadcasts the decision to other vehicles.

ML-Based DETECTION

The dataset, called Vehicular Reference Misbehavior Dataset (VeReMi), is used as a basis to assess the performances of new detection approaches.

ML-based techniques learn the behavior of attacks and classify vehicles as normal or misbehavior/attacker.

The performance of the approach is evaluated in terms of detection rate e.g. true positive rate, false positive rate, and performance indicators generated by these rates i.e. precision, recall, accuracy, and F1-score.



IDEATION

To filter out the attacker among all the users , we have used supervised machine learning.

A distributed ML-based intrusion detection scheme is developed exploiting these features for position falsification attacks in VANETs

To get the positive result , we have used classifications like Random Forest , Logistic Regression and K-Nearest Neighbors.

Hence, an Ensemble Learning method is proposed to carry out the proposed feature combinations.



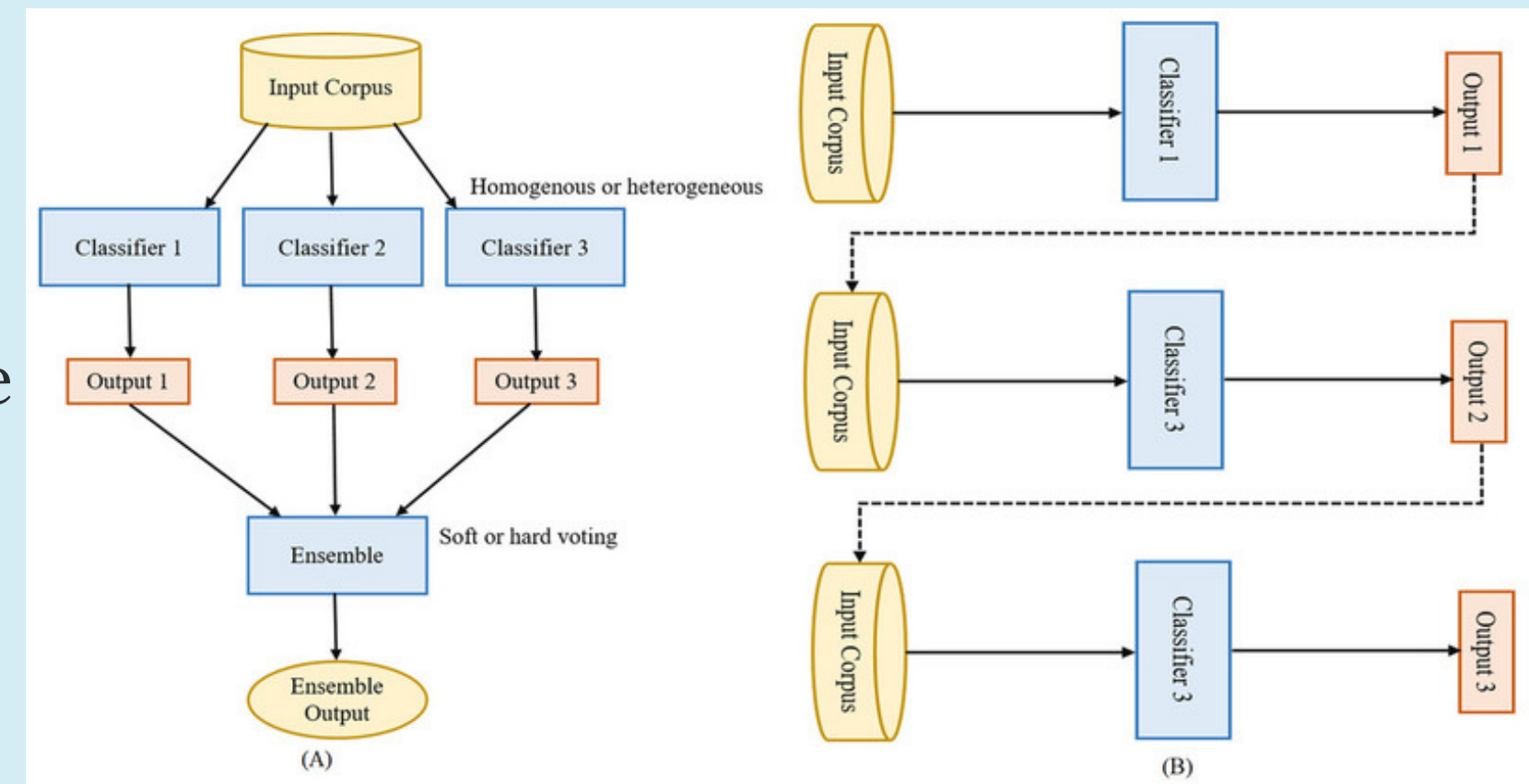
What is Ensemble Learning ?

Ensemble learning is a general meta approach to machine learning that seeks better predictive performance by combining the predictions from multiple models.

Although perhaps non-intuitive, more random algorithms (like random decision trees) can be used to produce a stronger ensemble than very deliberate algorithms (like entropy-reducing decision trees).

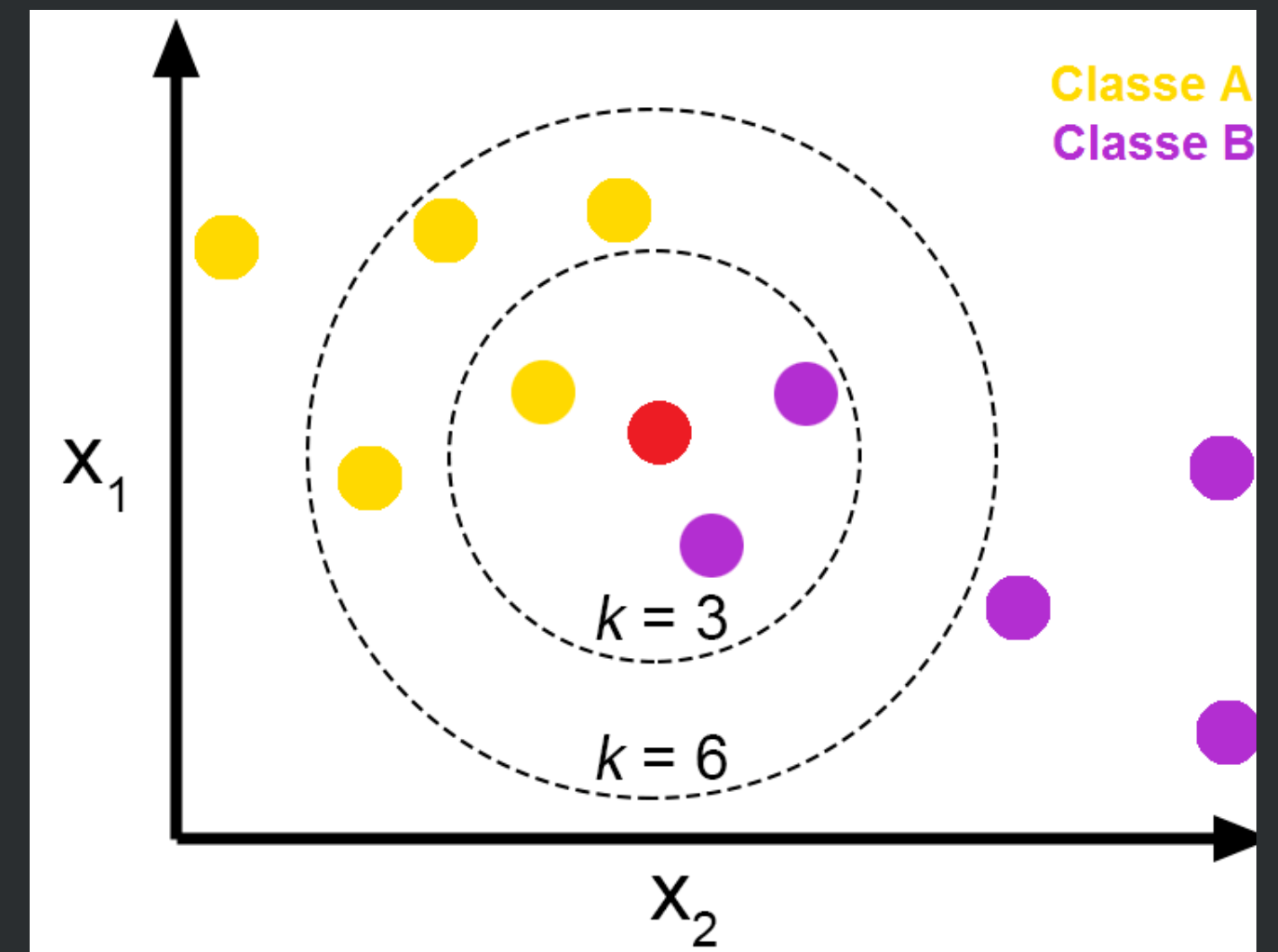
Some of common types of Ensembles are:

- Bayes Optimal Classifier
- Bagging
- Boosting
- Bayesian model combination
- Bucket of models
- Stacking



What is K-NN ?

- K-Nearest neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.



What is RFF ?

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique.

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.

The Working process can be explained in the below steps and diagram:

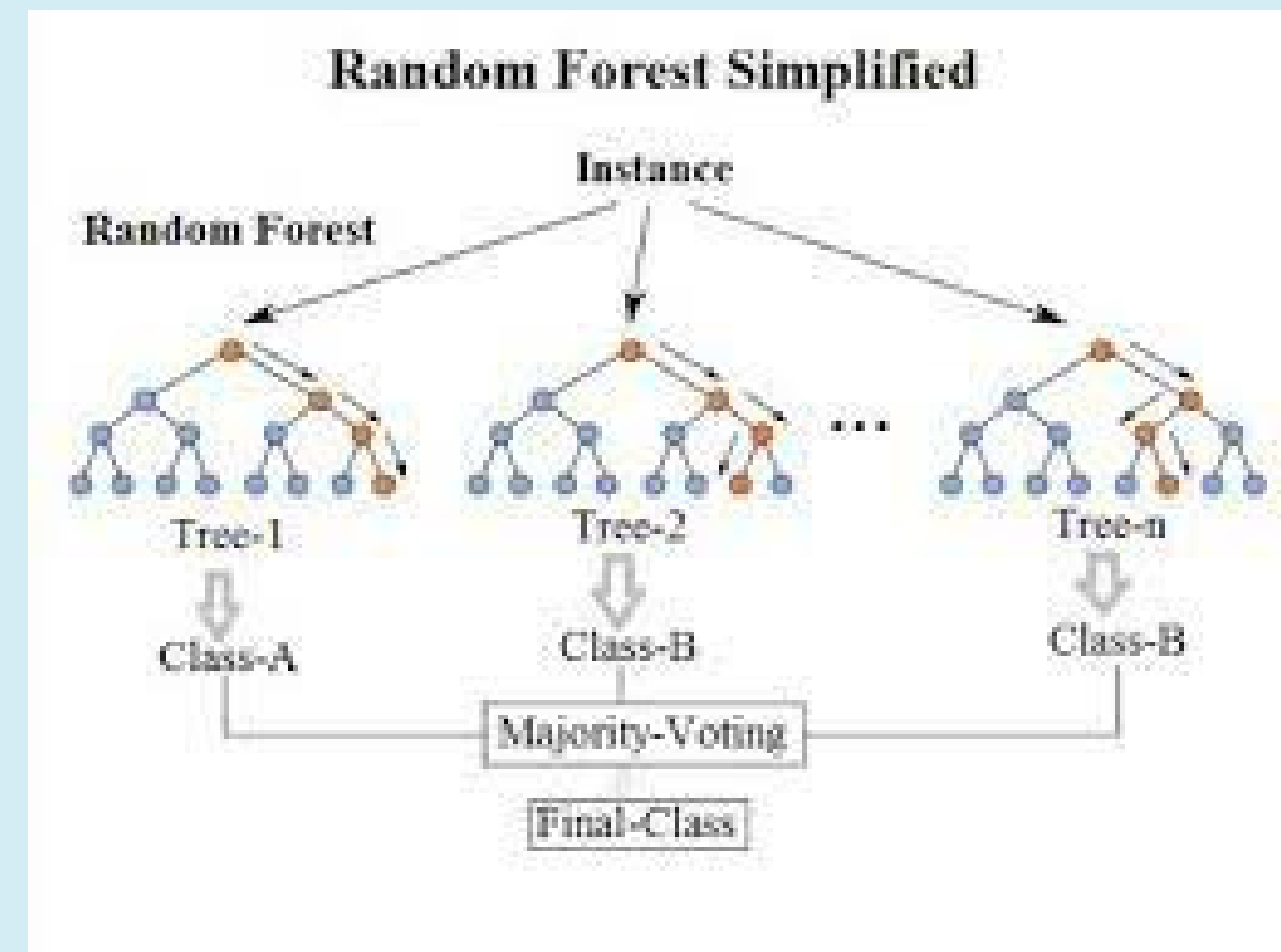
Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.



Extracting Features

To detect position falsification attacks, the Position (P) and Speed (S) are considered in this work for both the receiver (R) and the sender (S).

Moreover, the difference in position and speed between the last two BSM of the same sender are useful features confirmed by the previous works.

Now we make 6 equations to calculate the following :

1. Difference in position of same user.
2. Difference in speed of the same user.
3. Current position of same user.
4. Current speed of same user.
5. Distance between sender and receiver.
6. Difference between speed of sender and receiver.

$$\Delta P_{S,a}(i) = P_{S,a}(i) - P_{S,a}(i - 1)$$

$$\Delta S_{S,a}(i) = S_{S,a}(i) - S_{S,a}(i - 1)$$

$$\Delta P_{R,a}(i) = P_{R,a}(i) - P_{R,a}(i - 1)$$

$$\Delta S_{R,a}(i) = S_{R,a}(i) - S_{R,a}(i - 1)$$

$$\Delta P_{R \rightarrow S,a} = |P_{R,a} - P_{S,a}|$$

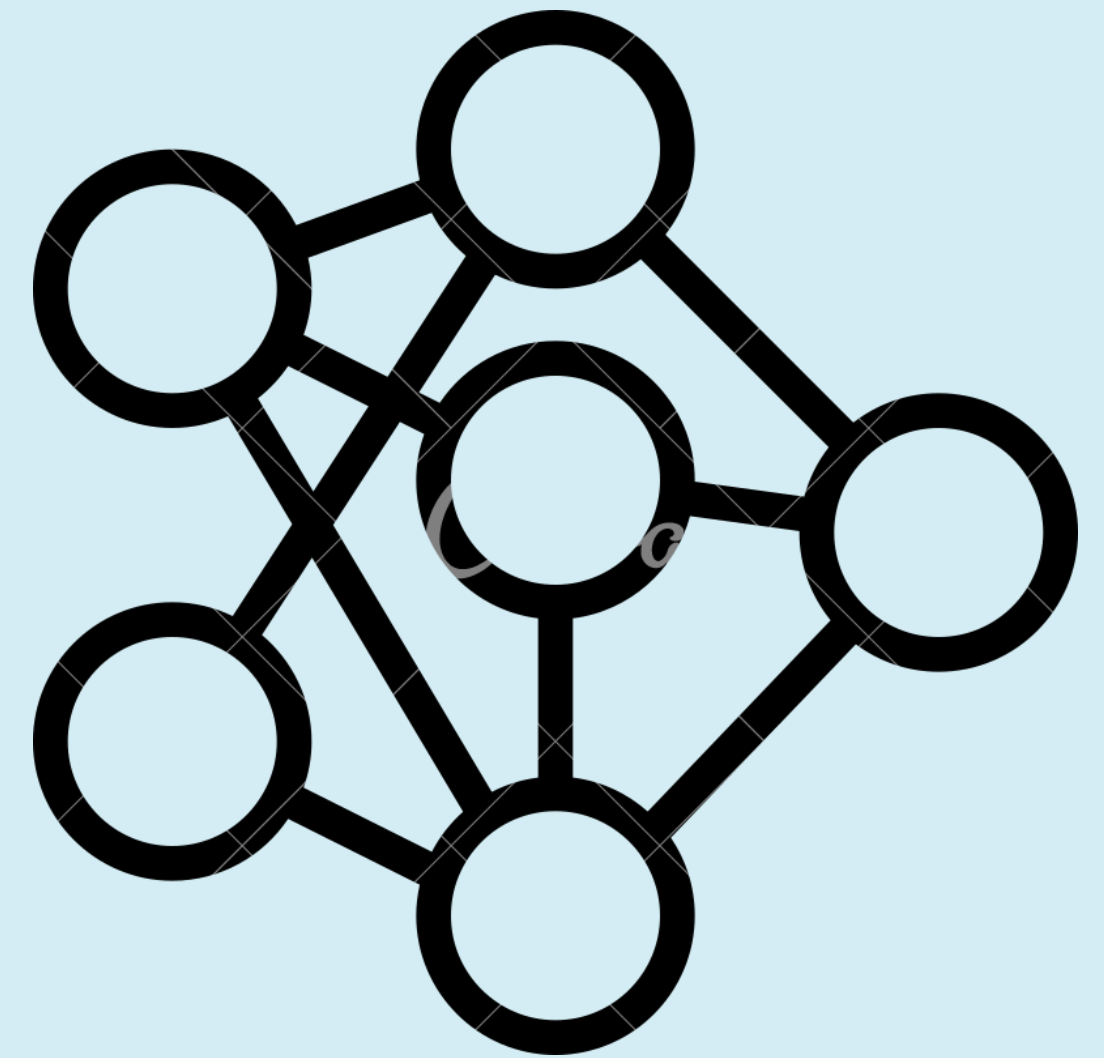
$$\Delta S_{R \rightarrow S,a} = |S_{R,a} - S_{S,a}|$$

ALGORITHM

So to check the status of the user , we are making features from the given VeReMi dataset and returning a Boolean value , where 0 denotes that the vehicle is a normal vehicle and 1 denotes that the vehicle is an attack vehicle.

We are dividing the dataset into 2 types , Training and Testing. First we are applying our classification algorithms of RF , K-NN , and Logistic Regression separately to train the model. Then the testing part is done. After testing , analysis of the result obtained is done by plotting graphs.

To get the result , we are using the concept of Ensemble learning , i.e. stacking of the result obtained from all 3 classifications is done to improve the overall accuracy.



How Features are Extracted

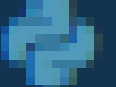
Firstly , we have a big dataset which contains raw JSON files which contains data in a JSON format.

Now this data needs to be converted into a usable csv file , so for that we wrote a script under file name "data_to_csv" which converts json data into a formatted csv files with 16 columns and rows equals to that of number of data.

Features are generated by subtracting and computing the values of these columns to get difference between speed of sender and receiver and difference between positions of same user and that of other user as well.

STEPS

```
✓ data
  ⌵ AttackerType1-star...
  {} GroundTruthJSONI...
  {} JSONlog-0-7-A0.js...
  {} JSONlog-1-13-A0.j...
  {} JSONlog-2-19-A0.j...
  {} JSONlog-3-25-A0.j...
  {} JSONlog-4-31-A0.j...
  {} JSONlog-5-37-A0.j...
```

 `data_to_csv.py`

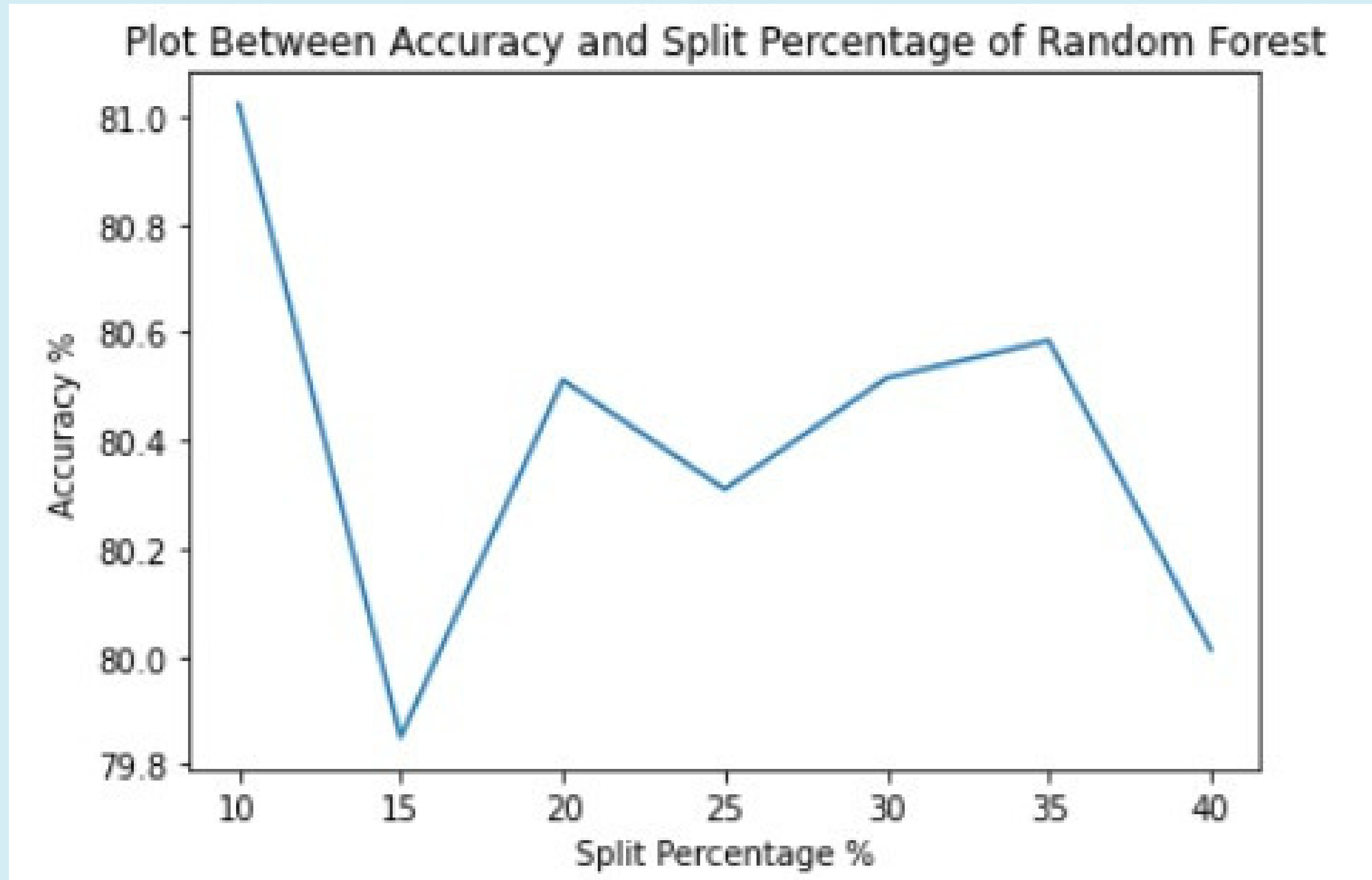
```
type
rec_tim
send_time
sender_id
msg_id
RSSI
sender_p_x
sender_p_y
sender_spd_x
sender_spd_y
rec_p_x
rec_p_y
rec_spd_x
rec_spd_y
rec_id
attacker_type
```

```
30 frames1=[atk,atk1,atk2]
31 attack=pd.concat(frames1,axis=1)
32 attack
33
34 attack.drop(['pos','spd'], inplace=True, axis=1)
35 attack
36 # attack.drop(attack.index[[0]])
37 tmp_Attack = attack
38
39 # for i in range(tmp_Attack.shape[0]):
40 i=0
41 while i<=attack.index[-1]:
42     # print(f'{attack["type"][i]} {attack["type"][i+1]}')
43     attack.index = range(len(attack))
44     print(i)
45     print(attack.index)
46
47     if(i==attack.index[-1] and attack.at[i,"type"]==2):
48         attack = attack.drop(i)
49         break
50     # print("ccc")
51     # print(attack.index)
52
53     elif(attack.at[i, "type"] == 2 and attack.at[i+1, "type"] != 3):
54         attack = attack.drop(i)
55         print("dropped")
56         # i+=1
57     else:
58         # print(attack.shape)
59         i+=1
60         # print("hello")
61 # attack = attack.drop(attack.index[[0]])
62 # print(attack.shape)
63 bsm = attack
64 bsm
65
66 # print("byeeee")
67
68 bsm['spd-x'] = np.round(bsm['spd-x'], decimals = 2)
69 bsm['spd-y'] = np.round(bsm['spd-y'], decimals = 2)
70 bsm['pos-x'] = np.round(bsm['pos-x'], decimals = 2)
71 bsm['pos-y'] = np.round(bsm['pos-y'], decimals = 2)
72 bsm['rec_pos_x'] = np.round(bsm['pos-x'], decimals = 2)
73 bsm['rec_pos_y'] = np.round(bsm['pos-y'], decimals = 2)
74 bsm['rec_spd_x'] = np.round(bsm['spd-x'], decimals = 2)
75 bsm['rec_spd_y'] = np.round(bsm['spd-y'], decimals = 2)
76 bsm
```

Random Forest Implementation

```
def x_to_feat(X):  
    return np.stack((X[:,1] - X[:,2], X[:,8] - X[:,12], X[:,9] - X[:,13])).T  
  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.datasets import make_classification  
import numpy as np  
my_data = np.genfromtxt('data.csv', delimiter=',') #skip first row as it is header  
np.random.shuffle(my_data)  
X = my_data[:, :-1]  
Y = my_data[:, -1]  
# feat_x = x_to_feat(X)  
print(X.shape)  
print(Y.shape)  
print(feat_x.shape)  
  
clf = RandomForestClassifier(n_estimators = 10, max_depth=5, random_state=0)  
clf.fit(feat_x, Y)  
  
X_eval = my_data[40:, :-1]  
print(X_eval.shape)  
feat_x_eval = x_to_feat(X_eval)  
print(feat_x_eval.shape)  
  
Y_eval = my_data[40:, -1]  
print(Y_eval.shape)  
# print(clf.predict(feat_x_eval))  
# print(Y_eval)  
# print(clf.predict(feat_x_eval)-Y_eval)  
print(clf.score(feat_x_eval, Y_eval))
```

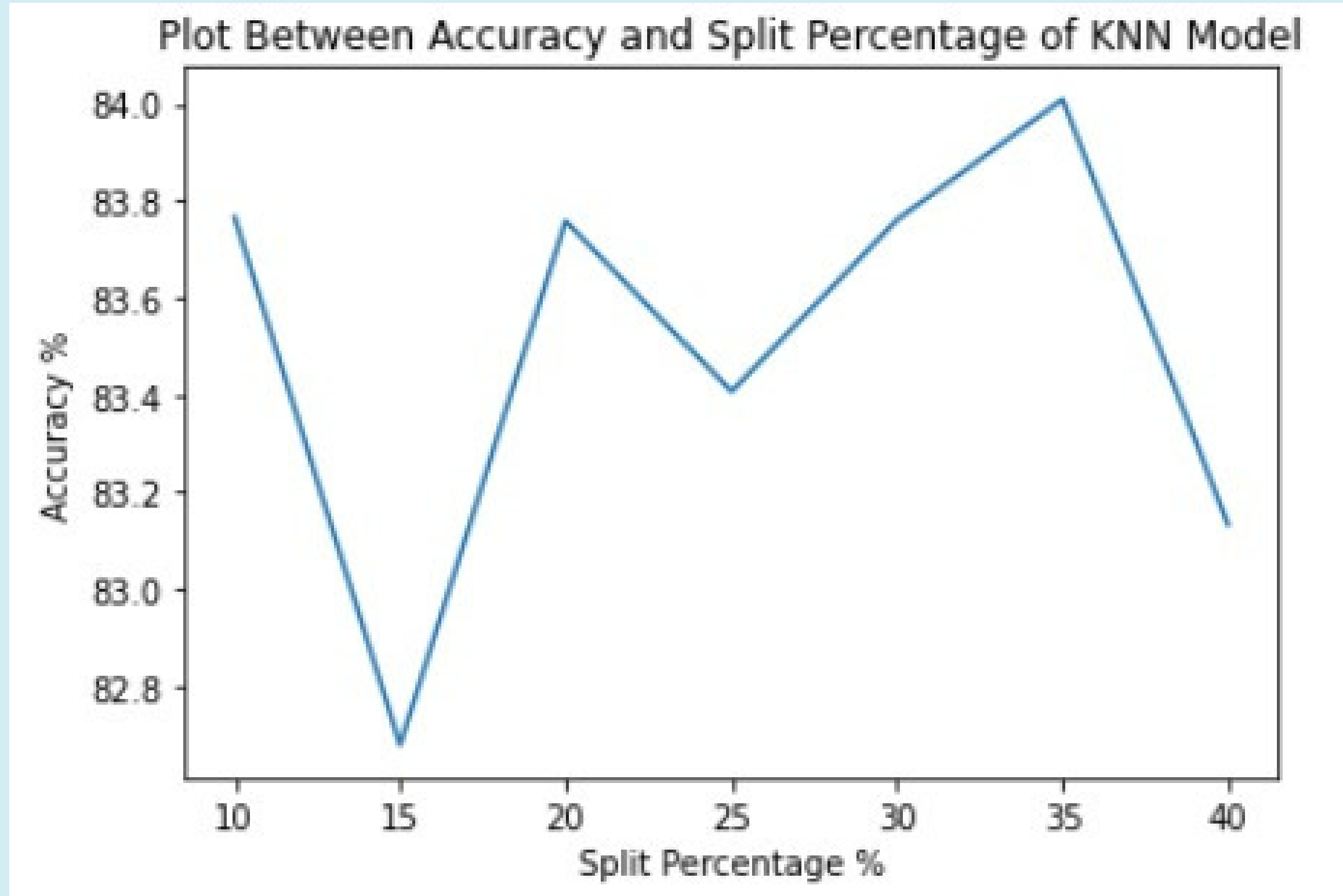
Graph Plotting



K-NN Implementation

```
import numpy as np
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
#create new a knn model
knn = KNeighborsClassifier()
#create a dictionary of all values we want to test for n_neighbors
params_knn = {'n_neighbors': np.arange(1, 25)}
#use gridsearch to test all values for n_neighbors
knn_gs = GridSearchCV(knn, params_knn, cv=5)
#fit model to training data
knn_gs.fit(X_train, y_train)
```

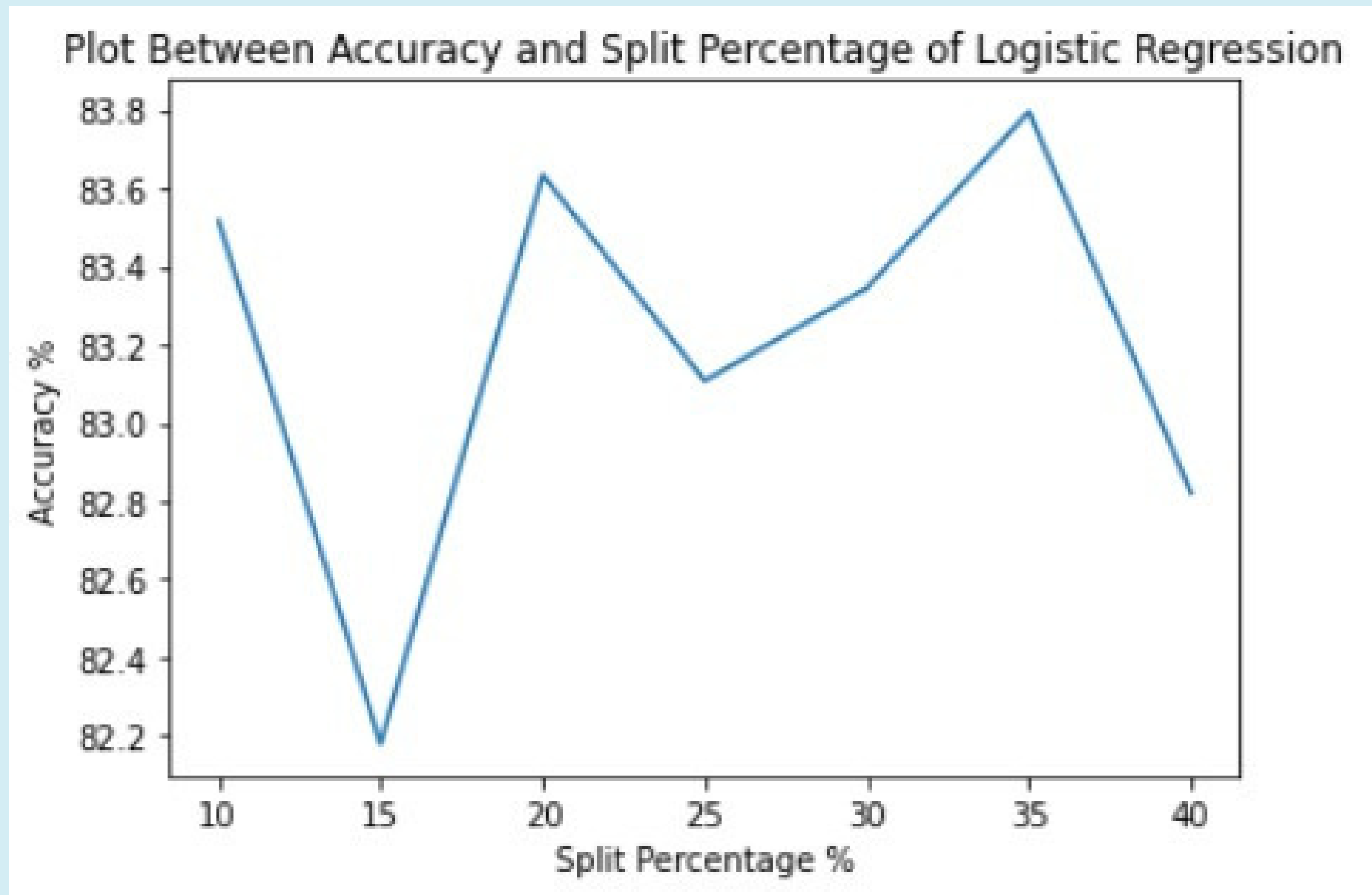
Graph Plotting



LR Implementation

```
from sklearn.linear_model import LogisticRegression
#create a new logistic regression model
log_reg = LogisticRegression()
#fit the model to the training data
log_reg.fit(X_train, y_train)
```

Graph Plotting

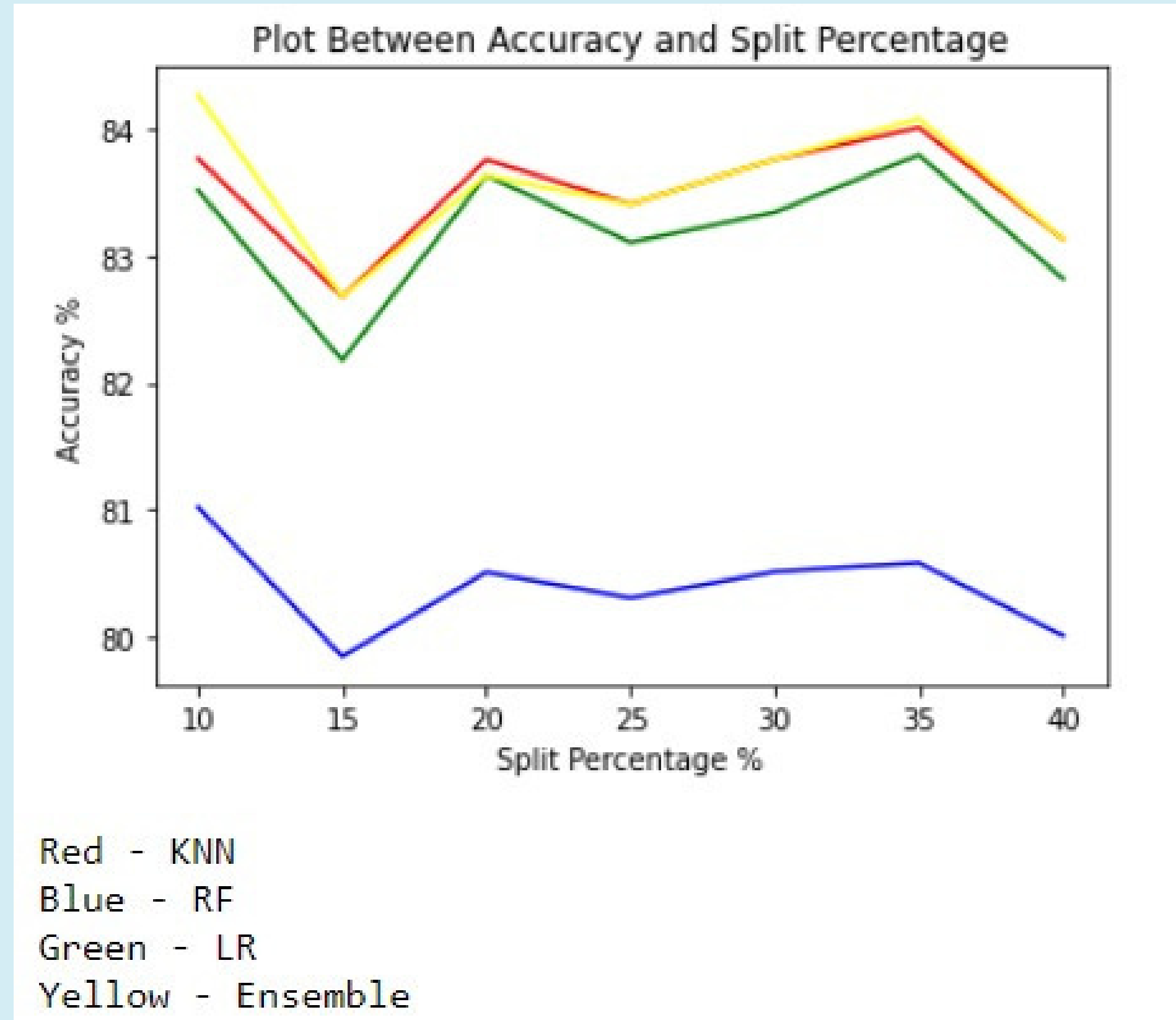


Ensemble Implementation

```
from sklearn.ensemble import VotingClassifier
#create a dictionary of our models
estimators=[('knn', knn_best), ('rf', rf_best), ('log_reg', log_reg)]
#create our voting classifier, inputting our models
ensemble = VotingClassifier(estimators, voting='hard')

#fit model to training data
ensemble.fit(X_train, y_train)
#test our model on the test data
ensemble.score(X_test, y_test)
```

COMPARE RESULTS



Future Insight

Currently, we worked on the dataset and model generation. We will provide a user interface to this model, so that users can interact with the app and we could take road safety to their door.

We will also work on our model to improve the accuracy a little more. We would also study different types of attacks to a vehicle and how to detect them as well.

There is a scope of taking a track to hardwares and creating a mock VANET attack environment. The boundary is limitless and we are the explorers.

THANKYOU

