Turkish Journal of Computer and Mathematics Education Vol.12 No. 7 (2021), 2097- 2109

Research Article

Comparative Analysis Of Classification Techniques Used In Machine Learning As Applied On A Three Phase Long Transmission Line System For Fault Prediction Using Python

Anjan Kumar Sahoo[1*], Abhinna Chandra Biswal[2]

[1] Research Scholar,Centurion University of Technology and Management Odisha & Assistant Professor, Electrical Engineering Department,College of Engineering Bhubaneswar, Odisha, India

[2] Professor, Electrical Engineering Department,Centurion University of Technology and Management, Odisha, India

Article History: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 16 April 2021

Abstract: The recent developments in the technology made by organizations have led to a quicker, simpler and a very accurate data analysis. The use of machine learning techniques have been exponentially increasing in the analysis of data in different fields ranging from medicine to defense, education, finance and energy applications. The machine learning techniques reduce further meaningful information processed by data mining. These significant and meaningful information help organizations to establish their future policies to get more advantages in terms of time and cost. In this paper the author has tried to present the best classification method by having a comparative analysis on various methods such as Logistic Regression, Support Vector Machine, Naïve Bayes and K-Nearest Neighbors etc. for a particular use case i.e. prediction and classification of transmission line faults. The author has made this analysis by utilizing both Python and MATLAB Simulink. This will surely help the researchers to know details such as accuracy,f1 score, mean square error etc of various classification methods.

Keywords: *Three Phase Fault, Logistic Regression, Support Vector Machine, Naïve Bayes, K-Nearest Neighbors and Line To Ground Fault.*

Introduction

Machine learning is a significant part of artificial intelligence with respect to current advancement in technology. It helps computers in predicting the future events and modeling based on provided data. Recently, the artificial intelligence frameworks are applied for the implementation of the fault location and fault type prediction algorithm to obtain the complex relationship between fault prediction[6,7] and feature extraction from faulty signals. The selection of proper machine learning tool for the classification and prediction of type of fault plays an important role. The accuracy and mean square error are to be noted individually for every classification tools. In this article, the author has taken a three phase long transmission line with pi network model and created variety of faults in that system. The features of both healthy and faulty system are extracted from MATLAB workspace. These extracted features are used as use case for the comparison of classification tools of machine learning[8].

With the innovation in technology the demand of electrical energy also increases rapidly. Hence transmission line faults are to be processed properly. Or else this may lead to poor reliability, power quality disturbances, large scale power outage etc. As per the statistics present in Table 1, the transmission line is mostly affected by single phase to ground fault i.e. 70%.Out of these, again 70-90% of the faults are arcing faults which can be predicted and protected by proper and suitable algorithms.

Table I. Relative probability of occurrence of fault in percentage

Type of Fault	L-G	L-L	L-L-G	L-L-L L-L-L-G L-L, L-G	Total
% of occurrence	70	15	10	5	100

As seen most of the HV faults occurred at transmission line are single phase to ground fault .The three phase faults happen very rarely. Hence the author has taken single phase to ground fault as the faulty condition only.

The 2nd part of the article describes about the proposed three phase transmission line system. The 3rd part narrates about the Logistic Regression. The 4th part explains about the Support Vector Machine. The 5th part describes about the Naïve Bayes classification. The 6th part explains the K-Nearest Neighbors algorithm. The 7th part presents MATLAB Simulation results and compares the various classification methods. The last part concludes the article.

Three phase transmission line system

As shown in fig.1,the author has simulated a high voltage three phase transmission line system over a length of 300km.Long transmission line with a voltage magnitude of 400kv(Ph-Ph rms) is taken only to have a lossless line. The fault is introduced intentionally at the mid of the transmission line. The voltage and current magnitudes before and after the fault are extracted from MATLAB simulation workspace. The author has taken single line to ground fault as the faulty condition. The phase a to ground fault is introduced at the mid length of the line. Hence the line voltages V_{ab} and V_{ca} are mostly affected by the fault. The individual line currents and line voltages are taken as the features. Total number of independent features used are six and the status(LG_fault/No_fault) of the transmission line is taken as dependent variable. In total the author has used these extracted features in 4 different machine learning such as Logistic Regression, Support vector machine, Naïve Bayes and K-Nearest Neighbor algorithms separately to know the most efficient and suitable prediction algorithm for the transmission line.

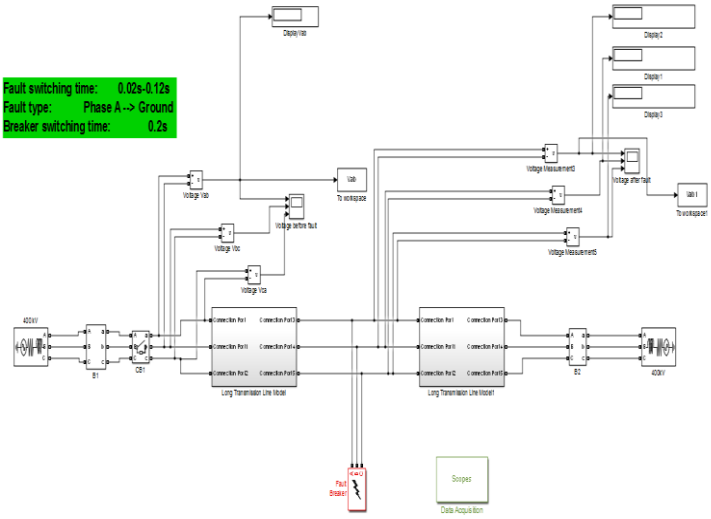


Figure 1. MATLAB simulation of three phase transmission line.

Logistic regression

It is very efficient and does not require any computational techniques. It is very easily interpretable and does not require any input features to be scaled. It is also quite easy to regularize. It does not require any tuning, it also outputs well calibrated predicted probabilities. Like linear regression, it works quite better when we remove attributes which are unrelated to the output variable and also attributes which are very similar to each other. Feature engineering plays an important role in regards to the performance of logistic regression. Its decision

surface is linear, hence it can't solve any non-linear problems. Apart from machine learning, it is much used in statistics, medical fields, social sciences etc. It is also called a statistic model. It is used to predict the probability of a certain class or event such as good/bad, yes/no, win/lose, pass/fail, alive/dead etc. The outcome must be discrete i.e. 0 or 1 and nothing in between. In machine learning, it is a supervised learning and output is always categorical type. Main aim of this method is to build a logistic relationship between independent and dependent variables. It is a sigmoid function and mathematically it can be derived from linear regression equation i.e. $Y=\beta_0+\beta_1X+\varepsilon$.

As the dependent variable in our use case is also binary, this method is suitable for our use case. The length of the dataset is 965.Total 80% of the dataset i.e. 772 data is used for training and 20% i.e. 193 data is used for testing purpose.

```
1 df.head()
```

	time	currentR	currentY	currentB	voltageRY	voltageYB	voltageBR	status
0	0.00000	-2.690268	-9.845947	12.536215	0.438299	-1.642281	1.203982	1
1	0.00003	-2.568361	-9.928361	12.496723	0.453767	-1.646374	1.192607	1
2	0.00006	-8.852052	-10.009838	12.456175	0.662349	-1.650321	0.987972	1
3	0.00009	3.605329	-10.090473	12.414474	0.662159	-1.654122	0.991963	1
4	0.00012	-9.228260	-10.170204	12.371678	0.655451	-1.657775	1.002325	1

Figure 2. Dataset representation through python program.

```
1 df.tail()
```

	time	currentR	currentY	currentB	voltageRY	voltageYB	voltageBR	status
960	0.01608	-13.080941	5.012832	8.068110	-1.403733	-0.129718	1.533451	0
961	0.01611	-13.063736	4.897531	8.166205	-1.394621	-0.145694	1.540315	0
962	0.01614	-13.045371	4.781795	8.263576	-1.385385	-0.161658	1.547042	0
963	0.01617	-13.025847	4.665634	8.360212	-1.376026	-0.177607	1.553632	0
964	0.01620	-13.005165	4.549059	8.456106	-1.366545	-0.193540	1.560085	0

Figure 3. Last 5 rows of dataset in python program.

The status is represented in binary form i.e. 1&0. LG_fault is represented by 1 and No_fault is represented by 0.As shown in figs 2-3, the python program shows both type of status.

```
print(classification_report(y_test,predictions))
print(accuracy_score(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.93      1.00      0.96       113
           1       1.00      0.89      0.94        80

    accuracy                           0.95       193
   macro avg       0.96      0.94      0.95       193
weighted avg       0.96      0.95      0.95       193

0.9533678756476683
```

```
y_pred=model.predict(x_test)
```

```
y_test.head()
```

```
889    0
468    0
168    1
405    1
70     1
Name: status, dtype: int64
```

```
y_pred[5]
```

```
1
```

```
from sklearn.metrics import mean_squared_error
```

```
mean_squared_error(y_test,y_pred)
```

```
0.046632124352331605
```

Figure 4. Accuracy report and mean squared error report in python program.

The classification report shown in fig.4 is found to have a very good precision, f1_score and accuracy_score. The mean squared error is only 4.67%.This implies that the prediction made by logistic regression can have an error of 4.67%.However this is acceptable.

```
y_pred[50]
```

```
0
```

```
y_pred[191]
```

```
1
```

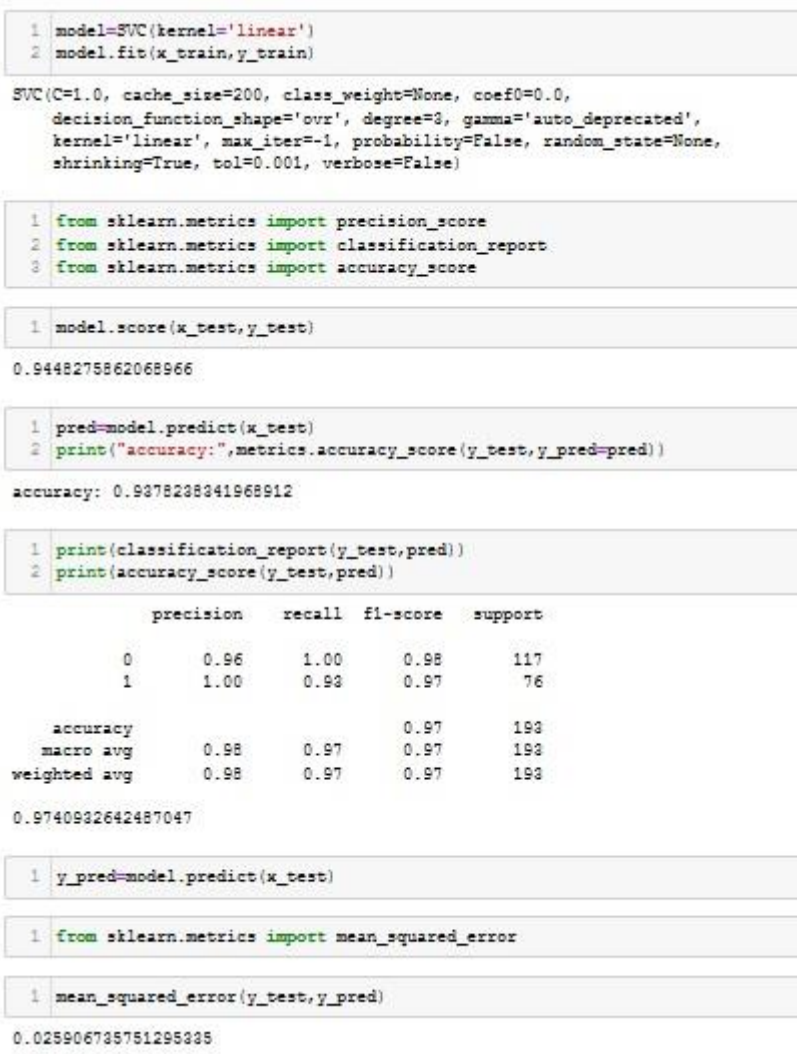
Figure 5. Fault prediction result of two random data in python program.

Fig.5 shows that most of our status predictions by using logistic regression come true.

Support vector machines

It is one of the most robust prediction algorithms. It is a supervised machine learning algorithm[1]. It constructs a hyper plane or a set of hyper planes in a high or infinite dimensional space that can be used for classification, regression etc. SVMs are useful in classification of images, texts and hypertext categorization, classification of satellite data like SAR data using supervised support vector machines. These are widely used in biological and other sciences. Apart from linear problems, the SVMs are also helpful for non-linear problems using kernel functions[2]. The author has used the python program to process the use case towards transmission line fault prediction. The total number of independent features, dependent variables and the length of data remain same as before.80% of the total data for training and 20% for testing purposes are used. The suitable programming results a better classification report in SVM. There are varieties of kernel functions. But here the author has used linear

kernel function for this use case. As shown in fig.6, the classification report is better than that of the previous algorithm. The accuracy score is 97.4% and mean squared error is only 2.6%.



```
model=SVC(kernel='linear')
model.fit(x_train,y_train)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='linear', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

```
from sklearn.metrics import precision_score
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```

```
model.score(x_test,y_test)
```

```
0.9448275862068966
```

```
pred=model.predict(x_test)
print("accuracy:",metrics.accuracy_score(y_test,y_pred=pred))
```

```
accuracy: 0.9378238341968912
```

```
print(classification_report(y_test,pred))
print(accuracy_score(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       117
           1       1.00      0.93      0.97        76

    accuracy                           0.97       193
   macro avg       0.98      0.97      0.97       193
weighted avg       0.98      0.97      0.97       193

0.9740932642487047
```

```
y_pred=model.predict(x_test)
```

```
from sklearn.metrics import mean_squared_error
```

```
mean_squared_error(y_test,y_pred)
```

```
0.025906735751295335
```

Figure 6. SVM initialization, classification report and mean square error in python.

Naïve bayes algorithm

It is based on Bayes theorem that is used to solve classification problems by following a probabilistic approach[3,4]. It is based on the idea that the predictor variables in a machine learning model are independent of each other[5].

$$P(A|B) = P(B|A)P(A)/P(B) \quad (1)$$

P(A|B) : Conditional probability of event A occurring, given event B.
P(B|A) : Conditional probability of event B occurring, given event A.
P(A) : Probability of event A occurring.
P(B) : Probability of event B occurring.

The author has implemented this algorithm for the same use case with same number of dependent and independent features and predicted the test result in fig.7(a).

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```

model=GaussianNB()

```
model=GaussianNB()
```

```
model.fit(x_train,y_train)
```

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

```
predictions=model.predict(x_test)
print(predictions)
print(y_test)
```

```
[0 0 1 1 0 1 0 0 0 1 0 1 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0
 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 1 0 1 1
 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 1 0 0 1 0 1 1 0 1 1 0 0 0
 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 1 1 0 1 1 0 1
 0 1 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0 1 1 1 1 0 0 0 0 1 0 1 0 0 0 0 0 0 1
 0 0 1 0 1 0 0 0]
889    0
468    0
168    1
405    1
70     1
      ..
959    0
215    1
554    0
33     1
31     1
Name: status, Length: 193, dtype: int64
```

Figure 7(a). Gaussian function initialization, prediction result of test data.

```
print(classification_report(y_test,predictions))
print(accuracy_score(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.86      1.00      0.92       113
           1       1.00      0.76      0.87        80

    accuracy                           0.90       193
   macro avg       0.93      0.88      0.89       193
weighted avg       0.92      0.90      0.90       193

0.9015544041450777
```

```
y_pred=model.predict(x_test)
```

```
y_test.head()
```

```
889    0
468    0
168    1
405    1
70     1
Name: status, dtype: int64
```

```
y_pred[5]
```

```
1
```

```
from sklearn.metrics import mean_squared_error
```

```
mean_squared_error(y_test,y_pred)
```

```
0.09844559585492228
```

Figure 7(b). Prediction result of classification report, accuracy score and mean squared error

```
x_test[:5]
```

	time	currentR	currentY	currentB	voltageRY	voltageYB	voltageBR
889	0.01395	-11.354144	11.506184	-0.152040	-1.696667	0.950197	0.746469
468	0.00132	2.744213	-12.553353	9.809140	1.063243	-1.681196	0.617953
168	0.00504	23.167967	-8.667380	-4.289192	1.305687	-0.421395	-0.884293
405	0.02940	27.604932	8.023813	-13.088507	-0.266403	1.530356	-1.263953
70	0.00210	0.882934	-13.168269	7.373075	1.288565	-1.568603	0.280037

```
y_test[:5]
```

```
889    0
468    0
168    1
405    1
70     1
Name: status, dtype: int64
```

```
model.predict(x_test[:5])
```

```
array([0, 0, 1, 1, 0], dtype=int64)
```

```
model.predict_proba(x_test[:5])
```

```
array([[9.99888831e-01, 1.11169146e-04],
       [7.83990126e-01, 2.16009874e-01],
       [7.87242844e-02, 9.21275716e-01],
       [3.07576492e-05, 9.99969242e-01],
       [8.27762705e-01, 1.72237295e-01]])
```

Fig. 7(c) - Model prediction probability test report.

In this algorithm, we have used 80% of data for testing and 20% of data for training purpose like the previous algorithms. As it is observed in fig.7(b) that the accuracy result and mean squared error report of Naïve Bayes algorithm are 90.15% and 9.84% respectively.Fig.7(c) shows the model prediction probability. If we observe the x_test and y_test report then around 91% of our prediction matches with actual value and there is prediction mismatch of around 9%.

K-Nearest Neighbors algorithm

It is one of the simplest machine learning algorithms based on the supervised learning technique. It is a non-parametric algorithm as it does not make any prediction on underlying data. It is also under the category of lazy learner as it does not learn from the training set immediately rather it stores the dataset and at the time of classification[9], it performs an action on the dataset.

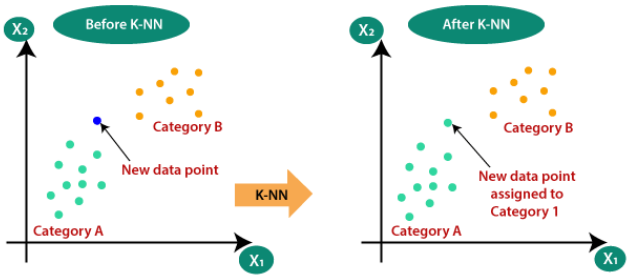


Figure 8. Classification of new data point using K-NN algorithm.

Suppose there are two types of data points in a dataset. The new data point can easily be categorized by using K-NN algorithm as shown in fig.8. There is no particular way to find the best value of K. Very low value of K is normally not acceptable as it may introduce noise or lead to the effect of outliers in the model. Although large values of K are good, it may find some difficulties.

The process of implementation is very easy and it is more effective if the training data is very large. It is very robust to the noisy training data. From all point of view, this method is found to be suitable for our

dataset. We have taken 80% of the data for training and 20% of the data for testing. The square root of the total number of testing data determines the value of K. Total number of testing data is 193.Hence the value of K taken for this use case is 13.This is necessary to mention that there is no such definite process to determine the value of K.

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

```
sc_x=StandardScaler()
x_train=sc_x.fit_transform(x_train)
x_test=sc_x.transform(x_test)
```

```
import math
math.sqrt(len(y_test))
```

```
13.892443989449804
```

```
len(y_test)
```

```
193
```

```
Classifier=KNeighborsClassifier(n_neighbors=13,p=2,metric='euclidean')
Classifier.fit(x_train,y_train)
y_pred=Classifier.predict(x_test)
```

Figure 9. Python program for initialization of K-NN function, assigning of K.

```
y_pred
```

```
array([0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,
       1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1,
       1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0,
       1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0,
       1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1], dtype=int64)
```

```
cm=confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[113   0]
 [  0  80]]
```

```
print(f1_score(y_test,y_pred))
```

```
1.0
```

```
print(accuracy_score(y_test,y_pred))
```

```
1.0
```

```
from sklearn.metrics import mean_squared_error
```

```
mean_squared_error(y_test,y_pred)
```

```
0.0
```

Figure 10. Python program showing predicted output, confusion matrix, accuracy score and mean squared error.

K-NN function is initialized in python environment as shown in fig.9.Without initialization; the classification process can't be done. Fig.10 shows the predicted output as 0 and 1.No_fault condition is denoted by 0 and LG_fault condition is denoted by 1.All these values completely match with our dataset. Hence accuracy score obtained is 100% and mean squared error is 0%. The confusion matrix is obtained to know how often the classifier is correct and how often it is wrong. To explain clearly, we have represented our confusion matrix in a detailed manner in Table 2.

Table II. Confusion matrix

N=193	Predicted:NO	Predicted:YES	
Actual: NO	TN=113	FP=0	113
Actual:YES	FN=0	TP=80	80
	113	**80**	

- There are two possible predicted classes: "yes" and "no". If we are predicting the faulty condition, for example, "yes" would mean there is LG_fault, and "no" would mean there is no fault.
- The classifier made a total of 193 predictions (e.g., 193 data are being tested for the faulty condition).
- Out of those 193 cases, the classifier predicted "yes" 80 times, and "no" 113 times.

In reality also, 80 data have the LG fault, and 113 data do not. The predicted value and actual value match as the accuracy score of the prediction is 100% or else both values don't match.

Let's know following terminologies which are required for the accuracy and precision calculation.

- **true positives (TP):** These are cases in which we predicted yes (there is fault), and there is actually fault.
- **true negatives (TN):** We predicted no, and there is no fault actually.
- **false positives (FP):** We predicted yes, but there is no fault actually. (Also known as "Type I error.")
- **false negatives (FN):** We predicted no, but there is fault actually . (Also known as "Type II error.").

The followings can be computed from confusion matrix for K-NN classifier:

- **Accuracy:** Overall, how often is the classifier correct?
  - (TP+TN)/total = (80+113)/193 = 1
- **Misclassification Rate:** Overall, how often is it wrong?
  - (FP+FN)/total = (0+0)/193 = 0
  - equivalent to 1 minus Accuracy
  - also known as "Error Rate"
- **True Positive Rate:** When it's actually yes, how often does it predict yes?
  - TP/actual yes = 80/80 = 1
  - also known as "Sensitivity" or "Recall"
- **False Positive Rate:** When it's actually no, how often does it predict yes?
  - FP/actual no = 0/113 = 0
- **True Negative Rate:** When it's actually no, how often does it predict no?
  - TN/actual no = 113/113 =1
  - equivalent to 1 minus False Positive Rate
  - also known as "Specificity"
- **Precision:** When it predicts yes, how often is it correct?
  - TP/predicted yes = 80/80 = 1
- **Prevalence:** How often does the yes condition actually occur in our data?
  - actual yes/total = 80/193 = 0.41

Result & Discussion

As discussed in previous section, we have simulated a high voltage three phase long transmission line over 300 km. using MATLAB. The fault[10] is introduced at the mid length of the line. Fig.11 shows the waveform of three phase line voltage without any fault.Fig.12 shows the waveform of three phase line current without any fault. The complete system is simulated after the occurrence of the LG(Phase A to ground) fault. The corresponding waveform is presented in fig.13.It is clearly observed that the line voltage V_{ab} and V_{ca} are affected but line voltage V_{bc} is not affected at all as the fault is from Phase A to ground.

However the author has collected all the fault related information in the required form and implemented most popular four different machine learning classification algorithms to know most suitable algorithm from accuracy and precision basis. For the ease of analysis, the author has made a clear comparative representation in tabular form in Table 3 and graphical form in fig.14.From both type of representation, it is crystal clear that all methods are quite good but k-NN method is most suitable for our dataset. It gives 100% accurate prediction.

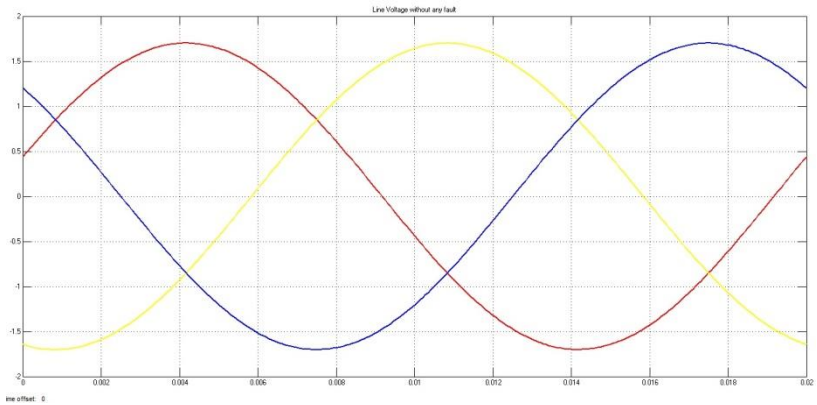


Figure 11. Three phase line voltage waveform without any fault.

Figure 12. Three phase line current waveform without any fault.

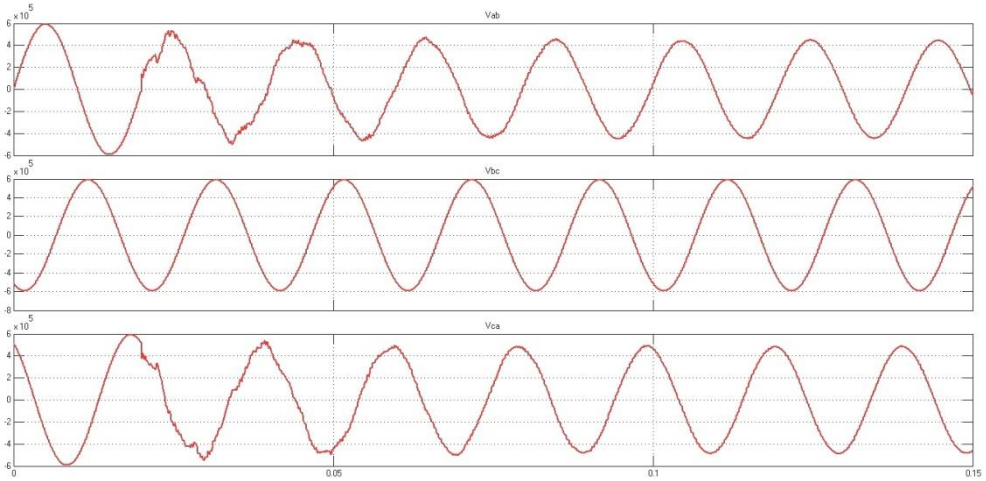


Figure 13. Three phase line current waveforms during LG fault.

Table III. Result analysis of classification algorithms.

Algorithm/Results	**Logistic Regression**	**Support Vector Machine**	**Naïve Bayes**	**K-NN**
Correctly classified	184	188	174	193
Incorrectly classified	09	05	19	0
Accuracy	95.33	97.4	90.15	1.0
Mean Squared Error	4.6	2.6	9.85	0.0

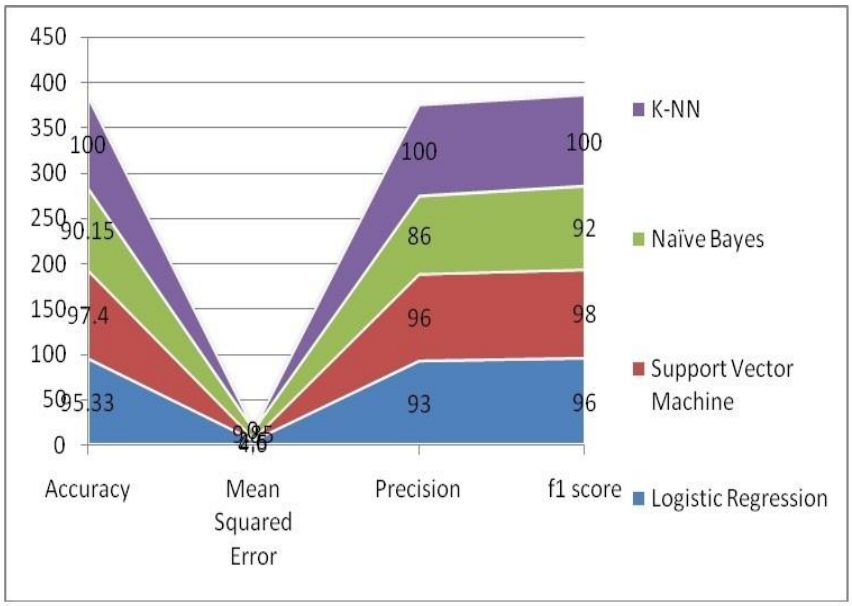


Fig. 14 - Graphical result analysis of classification algorithms

Conclusion

These days it is quite unavoidable to use soft computing due to rapid increase of amount of data to be processed in parallel. By using different machine learning techniques, it is quite easy to make accurate prediction about the future results to the data made available for processing. In this article, the author used various classification techniques on three phase long transmission line fault data and concluded that most appropriate algorithm for the classification of fault data is k-NN algorithm. It is called k Nearest Neighbor algorithm. This study will definitely be useful for different organizations and individuals working in all such related areas.

Acknowledgements

The authors would like to thank Centurion University for providing necessary facilities for doing this research. The authors are grateful to their colleagues, research scholars, friends and family for their support in all respect.

Conflict of interests

The authors declare no conflict of interest.

References

1. Halima ELAIDI, Younes ELHADDAR, Zahra BENABBOU and Hassan ABBAR (2018). An idea of a clustering algorithm using support vector machines based on binary decision tree. IEEE.
2. Kavitha S, Varuna S, and Ramya (2016). A Comparative Analysis on Linear Regression and Support Vector Regression. International Conference on Green Engineering and Technologies.
3. Ali Haghpanah jahromi and Mohammad Taheri (2017). A non-parametric mixture of Gaussian naive Bayes classifiers based on local independent features. Artificial Intelligence and Signal Processing Conference, pp. 209-212.
4. Ram, C. Sunitha, and R. Ponnusamy. "An effective automatic speech emotion recognition for Tamil language using Support Vector Machine." 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT). IEEE, 2014.
5. Kalyan Netti and Dr. Y Radhika (2015). A Novel Method for Minimizing Loss of Accuracy in Naive Bayes Classifier. IEEE International Conference on Computational Intelligence and Computing Research.
6. Murat, N. (2007) The Use Of Bayesian Approaches To Model Selection, M.Sc. Thesis, Ondokuz Mays University,Samsun, Turkey.
7. Bülbül, H.l, Ünsal, Ö.(2010) Determination of Vocational Fields With Machine Learning Algorithm, The Ninth International Conference on Machine Learning and Applications , IEEE Computer Society,710-713 Washington D.C.
8. Chandra, Dimple, and Partibha Yadav. "Prediction Of Software Maintenance Effort On The Basis Of Univariate Approach With Support Vector Machine." International Journal Of Computer Science And Engineering (Ijcse) 3.3 (2014): 83-90.
9. Witten, I.H, Frank, E. (2005) Practical Machine Learning Tools and Techniques Second Edition, Morgan Kaufmann Publications, USA
10. Anjan kumar sahoo and Abhinna chandra biswal et al.(2020) A Review on Machine Learning Applications in Transmission Line Fault Analysis. Indian Journal of Natural Sciences, 10(60).
11. Gopakumar, C., and S. Reshma. "Wavelet Based Analysis of ECG Signal for the Detection of Myocardial Infarction Using SVM Classifier." International Journal of Electronics and Communication Engineering 4.4 (2015): 9-16.

12. Shaoning Pang, Seiichi Ozawa and Nikola Kasabov (2005). Incremental Linear Discriminant Analysis for Classification of Data Streams. IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, 35(5), pp. 905-914.
13. Debosmita Chakraborty, Ujjal Sur and Pradipta Kumar Banerjee (2019) Random Forest Based Fault Classification Technique for Active Power System Networks. 5th IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE), pp. 1-4.
14. Ram, C. Sunitha, and R. Ponnusamy. "Recognising and classify Emotion from the speech of Autism Spectrum Disorder children for Tamil language using Support Vector Machine." International Journal of Applied Engineering Research 9.24 (2014): 25587-25602.
15. Lad, M. A. N. I. S. H. A., VV SHETE, and SB SOMANI. "Soft computing approaches for hand gesture recognition." Int. J. Comput. Sci. Eng. Inf. Technol. Res 3.4 (2013): 55-58.
16. Danthala, S. W. E. T. H. A., et al. "Robotic Manipulator Control by using Machine Learning Algorithms: A Review." Int J Mech Prod Eng Res Dev 8.5 (2018): 305-310.