Using Data Mining to Improve Assessment of Credit Worthiness via Credit Scoring Models

EE 583-Pattern Recognition

Term Project

Prepared by: Erinç Barış KOÇ

ANKARA-2018

TABLE OF CONTENTS

1) INTRODUCTION

This pattern recognition project is developed for the generation of credit scoring models. The related paper is developed by Yap et al.(2011). They generated models for accepting or rejecting the client's credit. They stated that there exist two main class for credit scoring models. The former is "good credit" which is the credit will be repaid with high probability. The second one is "bad credit" that the client has high possibility not to pay the financial obligations of the credit. They state that credit scoring models can be also applied to insurance, real estate, telecommunication and recreational clubs to foresee late payments. In addition, the dataset that is used in the paper belongs to historical payment of monthly subscription from members of local recreation clubs. The main reason is that the real customer data doesn't be shared publicly due to privacy concerns. Monthly subscription and permeant membership fee is collected but there exist increasing number of defaulters so that financial activities of club are affected. Their objective is to determine credit valuableness and determination of possible default based on local recreation club dataset. The features of the club dataset are in the Table 1. The target variable is binary.

Table 1::List of variables in data set

Variable description	Role	Measurement Level
Gender	Input	Nominal
Age in years	Input	Interval
District of address	Input	Nominal
Occupation	Input	Nominal
Race	Input	Nominal
Marital status	Input	Nominal
Number of dependents	Input	Interval
Number of cars	Input	Interval
Work sector	Input	Binary
Defaulters/ non-defaulters	Target	Binary

Logistic regression, scorecard and decision tree model are generated in the paper. Total of 2765 observations are available in data set. The SAS is used to generate the models. The training and validation data sets are generated with following percentages 70-30%. The percentage of correct selection for credit scorecard model, decision tree and logistic regression is 72%, 71%, 71.9%.

The local recreation club dataset is not available. However, the different data set related with bank customer creditability is used. The name of the data set used is "german_credit" which is provided by Prof. Dr. Hans Hofmann from University of Hamburg. In addition, there exist similar logistic regression and decision tree analyzes applied to this data set at Penn State College. Therefore, it is decided to use "german_credit" data set to generate the credit scoring models in this project. The data set consist of binary and numerical features. Moreover, logistic regression and decision tree analyzes are also applied to data set in order to compare the results with paper presented. Matlab and Python is used to generate related models. Applied models can be seen in the Table 2. The attributes of the data set is added to appendix. The wider explanation related with the data set is add as an separate file.The Matlab algorithms are especially generated for this project. They are my effort to understand and implement pattern recognition algorithms. For the python algorithms, predefined libraries are used.

Table 2::Generated Models

Model	Programs Used
Gaussian Naïve Bayes, Feedforward Neural Networks, Support Vector Machine, PCA with Regression Models	Python
KNN, Modified KNN, Fuzzy KNN, Linear Discriminant Classification Tree	Matlab

2) APPLIED MODELS

All the models are tested with randomly created 70-30% training and test sets. Normalization or standardization of attributes are applied to prevent high values affect results dominantly. The correlation between the variables and target value are analyzed to investigate whether there exists high correlation between the variables and target or not. If there exists high correlation between one of the attributes and target, that attribute can be used in the algorithm as a clue to improve the classification or prediction results. The correlation plot can be seen in Figure 1. It seems there is only first attribute has good correlation with target value. This correction especially can be important for the distance based method. It will be applied in the KNN algorithm.

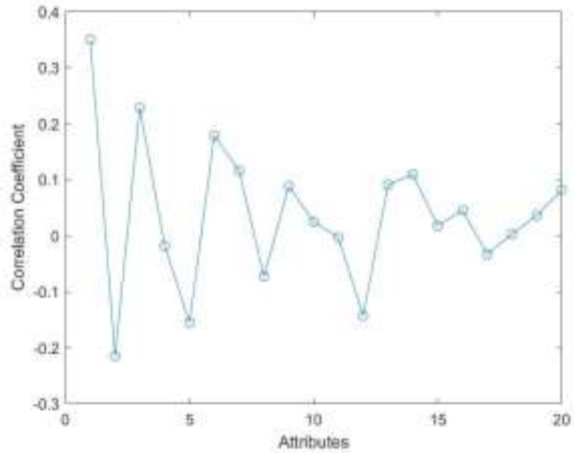

Figure 1:Correlation Coefficient

a) KNN Application with Modifications

The important parts of KNN algorithm are to choose correct the distance metric and number of neighbors. Therefore, Hamming and Jaccard distance are used separately for the binary attributes to analyze which distance is most appropriate for the purpose of the study. The Euclidean distance is used in the numerical attributes. The k value is tried between 1 and 100 to find the most appropriate k parameter in the beginning of the algorithm. The best ten k value based on the error percentages are selected and applied to test set. The KNN algorithm results are in Figure 2.

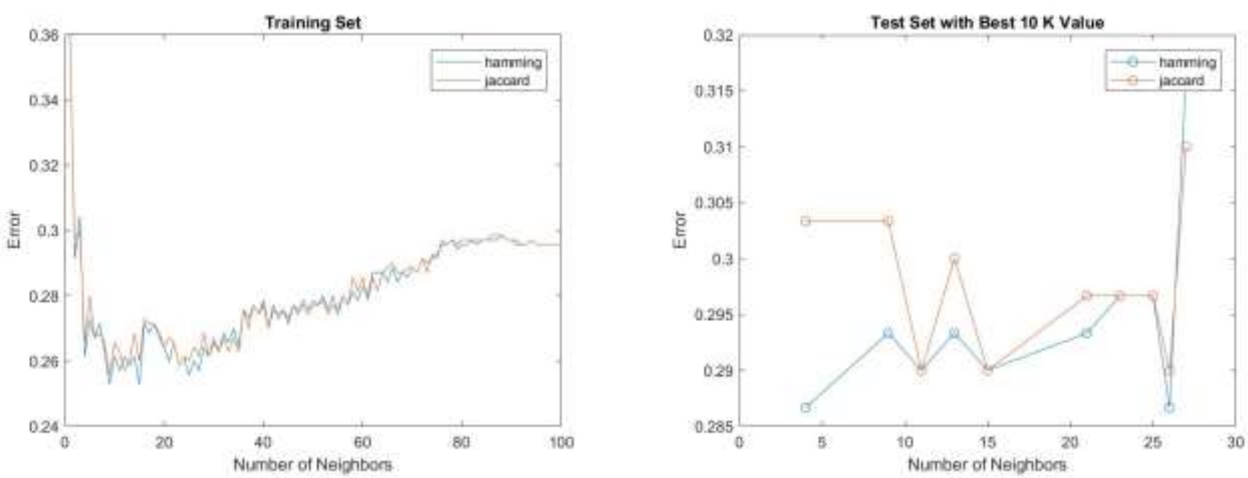

Figure 2:KNN algorithm results

The lowest possible error in the training set is 25%. However, this percentage increase in the test set even with best ten k values. The lowest error percentage in the test set is nearly 28.5%. In the standard KNN method, Hamming and Jaccard distance produce the similar results. There exists different modification of KNN in the literature. Modified KNN assign weight to k nearest neighbors according to their distance from the test point. It is also called the distance-weighted KNN algorithm. Weighted majority rule is applied in the algorithm. The weight function is defined as below.

$$w_j = \begin{cases} \dfrac{d_k - d_j}{d_k - d_1}, & if\ d_k \neq d_1 \\ 1, & if\ \ d_k = d_1 \end{cases}$$

The highest weight which is one is assigned to most closer neighbor. The weight reaches to zero for most distant customer. The results of modified KNN algorithm is Figure 3. The lowest error percentage is nearly 25% in both training and test set. The best k value is found as 15.

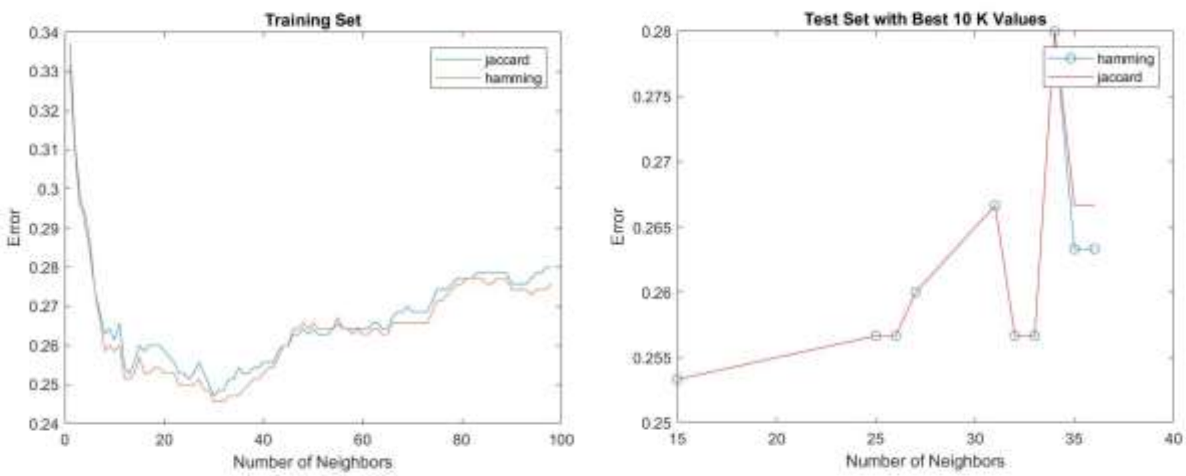

Figure 3:Modified KNN Algorithm Results

In fuzzy KNN algorithm, k nearest neighbors are found. Class membership function is assigned to each point. Test point is assigned to class which has highest membership function. The membership function can be explained as below.

$$\mu_i(p) = \frac{\sum_{j=1}^{K} \mu_{ij} \left(\dfrac{1}{d(P, X_j)^{\frac{2}{m-1}}} \right)}{\sum_{j=1}^{K} \left(\dfrac{1}{d(P, X_j)^{\frac{2}{m-1}}} \right)}$$

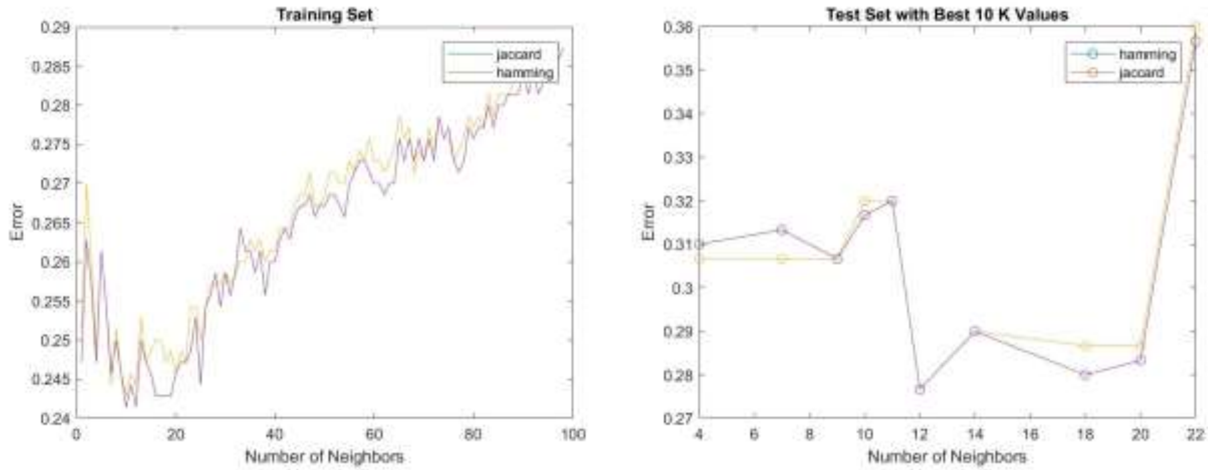

Figure 4:Fuzzy KNN Algorithm Results

b) Gaussian Naïve Bayes

The Gaussian Naïve Bayes algorithm is based on the assumption about independence of the predictors. The below equation is calculated and test point assigned to class which has higher probability.

$$P(Class|X) = \frac{P(X|Class)P(Class)}{P(X)}$$

$$P(Class|X) = P(x_1|Class) * P(x_2|Class) \dots . P(x_n|Class) * P(Class)$$

$P(x_1|Class)$ can be defined as the observing feature x_1 in a class. The application of Gaussian Naïve Bayes to categorical data is straight forward. However, there are different approaches for the application on the numerical data. The one way is to convert the numerical features into categorical one. The percentiles of the data can be used to convert numerical values into categorical ones. The other one is directly usage probability density functions. Theoretical distributions can be fit into numerical data and its probability density function are used to estimate above conditional probability. Different training and test sets are used to analyze performance. This method is analyzed with 100 different runs. The mean error percentage for training and test sets are 23.9% and 26.8%. The lowest error percentage in the training and test set is 23%.

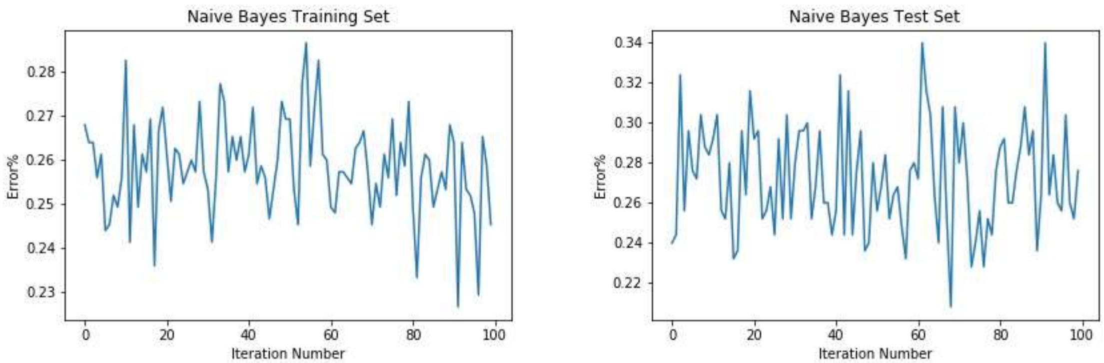

Figure 5: Gaussian Naive Bayes Algorithm Results

c) Linear Discriminant

100 iterations are tried with linear discriminants. The mean error for training and test set are 22.4% and 24.2%. The generated discriminant function is as below.

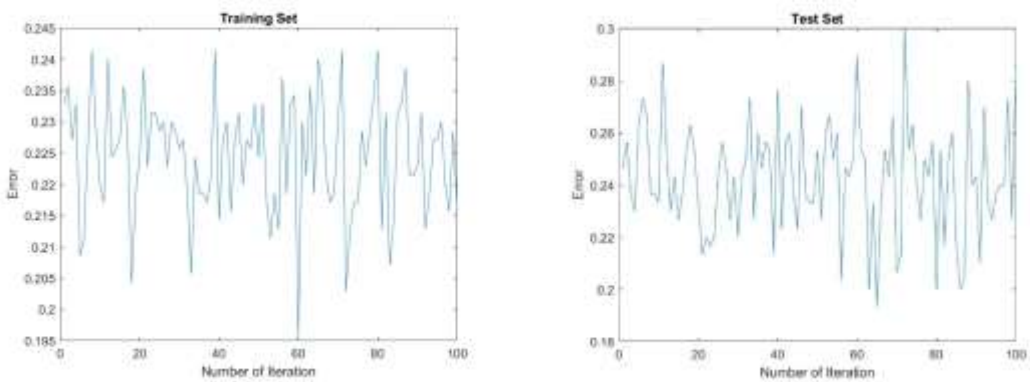

Figure 6: Linear Discriminant Algorithm Results

$$f(x) = \frac{1431 * x2}{100000} - \frac{65433 * x1}{100000} - \frac{38579 * x3}{100000} - \frac{553 * x4}{100000} + \frac{19 * x5}{100000} - \frac{5663 * x6}{20000} - \frac{13907 * x7}{100000} + \frac{35559 * x8}{100000} - \frac{43273 * x9}{100000} - \frac{9657 * x10}{25000} + \frac{3649 * x11}{50000} + \frac{4351 * x12}{20000} - \frac{547 * x13}{100000} - \frac{7569 * x14}{50000} - \frac{11729 * x15}{50000} + \frac{6319 * x16}{20000} - \frac{2133 * x17}{10000} + \frac{17949 * x18}{100000} - \frac{14033 * x19}{100000} - \frac{61927 * x20}{50000}$$

d) Principal Component Analysis(PCA) with Regression Models

Data is projected into lower dimensional space by using singular value decomposition method. The important part of the algorithm depends on the lower dimensional variables explains how much variance of original data set. The explained variance of origin data set with lower dimension variables are in Figure 7. The number of lower dimensional variables are tested between 1 and 20. Linear, logistic and polynomial regression is applied with PCA. The lower dimensional decomposition that provide the lowest error rate is applied to test set with all regression models. The polynomial regression produces the zero error after decomposition of 6 but it is overfitting of training set. The errors of the test set are higher in the polynomial regression.

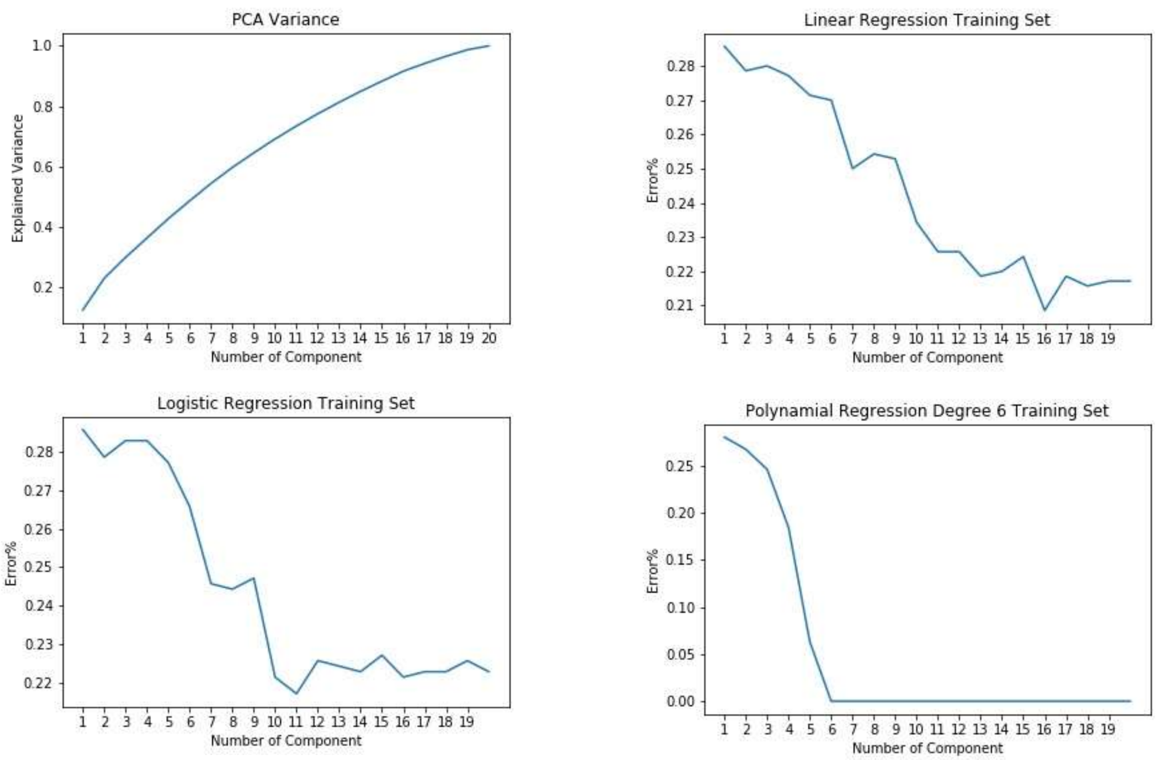

Figure 7:PCA Explained Variance and Algorithm Results for Training Set

Number of component that is used in test set for linear, logistic and polynomial regression are 16,11 and 6. Test set results are in Figure 8.

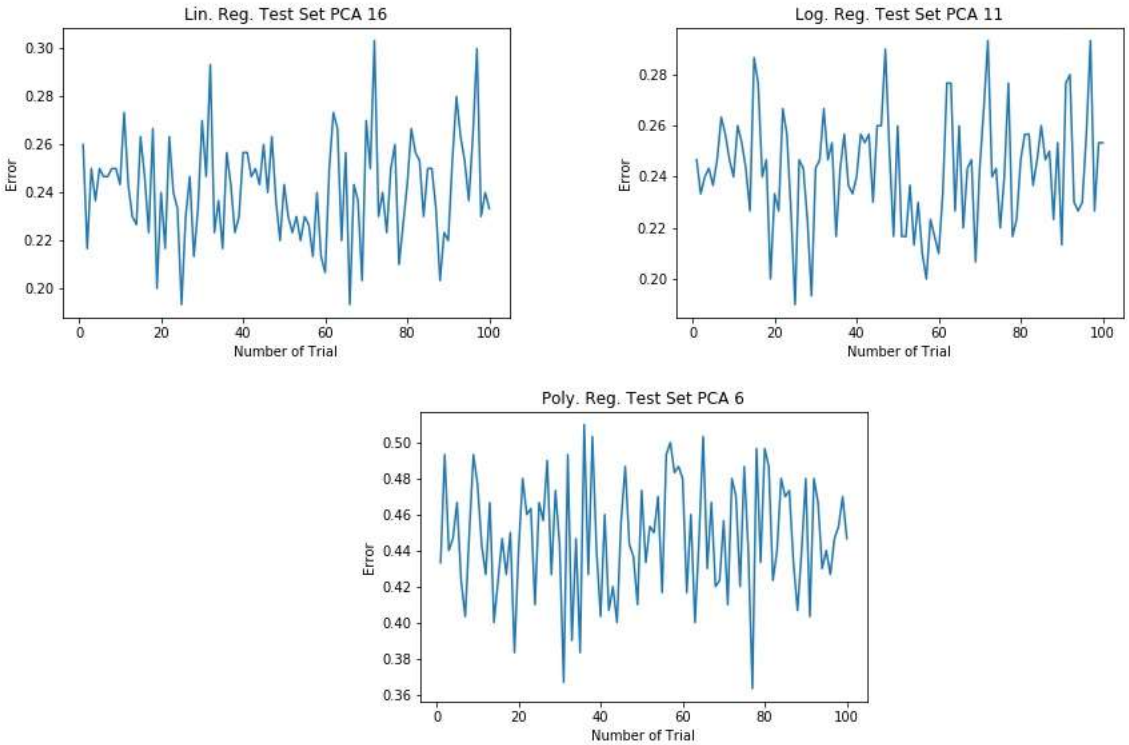

Figure 8:PCA Algorithm Results for Test Set

e) Support Vector Machine(SVM)

SVM are generally applied for supervised methods such as classification, regression and outlier detections. High dimensionality of data set lead us to apply SVM. SVM are applied with different combination of parameters. The results are in Table 3. All the related figures are added to appendix because of space concerns.

Table 3:SVM Algorithm Results

	Kernel Type Used	Mean Error of Training Set	Mean Error of Test Set
SVC	Polynomial	16.2%	70%
	Linear	22.5%	24.3%
	Radial Basis Function	13.8%	24.1%
	Sigmoid,	29.8%	26.2%
NuSVC (Upper Bound for Error 0.3)	Polynomial	38.9%	40%
	Linear	4%	27.7%
	Radial Basis Function	3%	26.6%

	Sigmoid,	61%	63.9%

f) Neural Networks

Feedforward neural networks are applied with different kind of activation functions and different number of hidden layers. The applied neural networks and their results are in the Table 4.

Table 4:ANN Algorithm Results

	Number of Hidden Layer	Dimension of Each Hidden Layer	Activation Function	Error Rate
1	3	10-10-5	Sigmoid	44.1%
2	3	20-20-10	Relu	42.5%
3	3	20-20-10	Relu, sigmoid	45.3%
4	3	20-20-20	Relu, sigmoid	46.3%
5	2	10-20	Relu, sigmoid	42.8%
6	2	10-10	Sigmoid	43.1%
7	2	20-10	Tanh sigmoid	41.4%
8	4	10-10-10-10	Relu, Tanh ,sigmoid	44.4%
9	4	10-10-10-10	Relu, sigmoid	44.1%

g) Classification Tree

The classification tree is generated to show the application at the main paper. The generated tree is pruned to prevent overfitting of training set. The cross validation method is used to determine the best pruned tree. The results are in the Figure 9. All the related results are added to appendix.

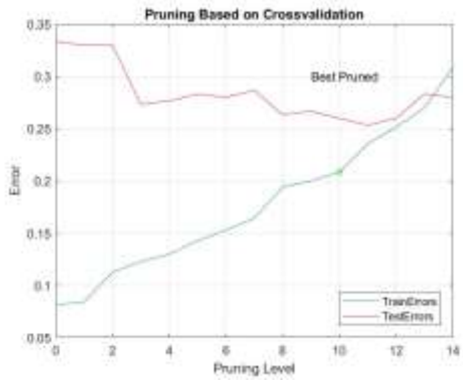

Figure 9:Classification Tree Algorithm Results

CONCLUSION

In the presented paper, they applied to logistic regression, score card model and classification tree. They stated that credit scoring models built using historical data of past applicants who were accepted could lead to a biased sample when used to evaluate new applicants. Their models are generated in SAS. It is also stated that there is no overall 'best' model. The performance of credit scoring models depends on the data structure, data quality and the objective of the classification. ANNs, MARS and SVM have shown only slight improvements in classification accuracy. In practical applications, classification methods which are easy to understand decision trees are more appealing to users. Error rate in test set is 72.0%, 71.2%, 71.9%.

In this project, wide range of pattern recognition algorithms are applied to data set. Lowest achieved error percentage is nearly 20% with PCA logistic regression and linear discriminant analyzes. Gaussian Naïve Bayes and KNN algorithms are most appropriate for this kind of purposes. Principal component analysis with regression equations provide good estimates. In ANN, the results are not in expected levels. The reason can be the overfitting of the training set. The decision tree and logistic regression is applied to compare the results in the presented paper. The accuracy in these two model is near to 78% with the best parameters. It is not actually appropriate to compare the results in the paper and this project because of the used data sets. In terms of run times, modified and fuzzy KNN algorithms took longest. As the data features are standardized in the algorithms, the results are not discussed by referring each attributes. However, the best parameters of each algorithm is presented in each section.

APPENDIX

1) German Credit Data Set Attributes

	Name of Attribute	Type	Levels
1	Status of existing checking account	qualitative	4
2	Duration in month	numerical	
3	Credit history	qualitative	5
4	Purpose	qualitative	11
5	Credit amount	numerical	
6	Savings account	qualitative	5
7	Present employment since	qualitative	4
8	Installment rate in percentage of disposable income	numerical	
9	Personal status and sex	qualitative	5
10	Other debtors / guarantors	qualitative	3
11	Present residence since	numerical	
12	Property	qualitative	4
13	Age in years	numerical	
14	Other installment plans	qualitative	3
15	Housing	qualitative	3
16	Number of existing credits at this bank	numerical	
17	Job	qualitative	4
18	Number of people being liable to provide maintenance for	numerical	
19	Telephone	qualitative	2
20	Foreign worker	qualitative	2

2) SVM Algorithm Results

Test set error rates for each SVM algorithm is in the Figure 10 and11.

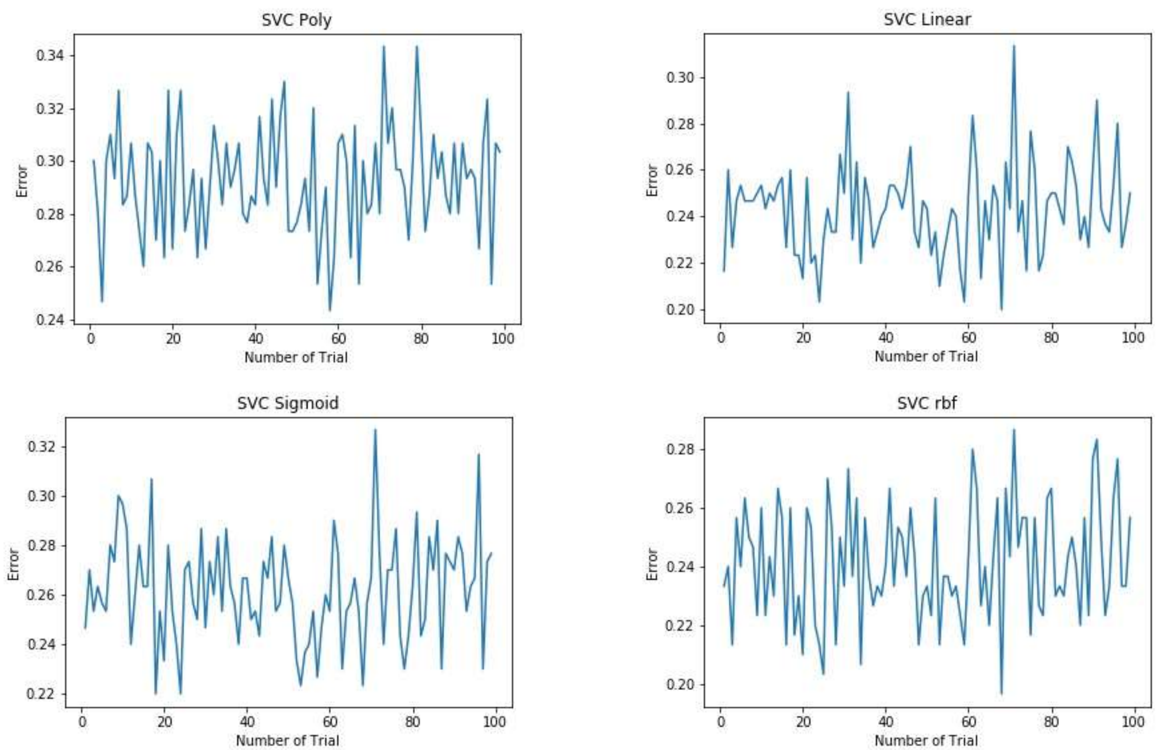

Figure 10: SVM Mean Test Set Error

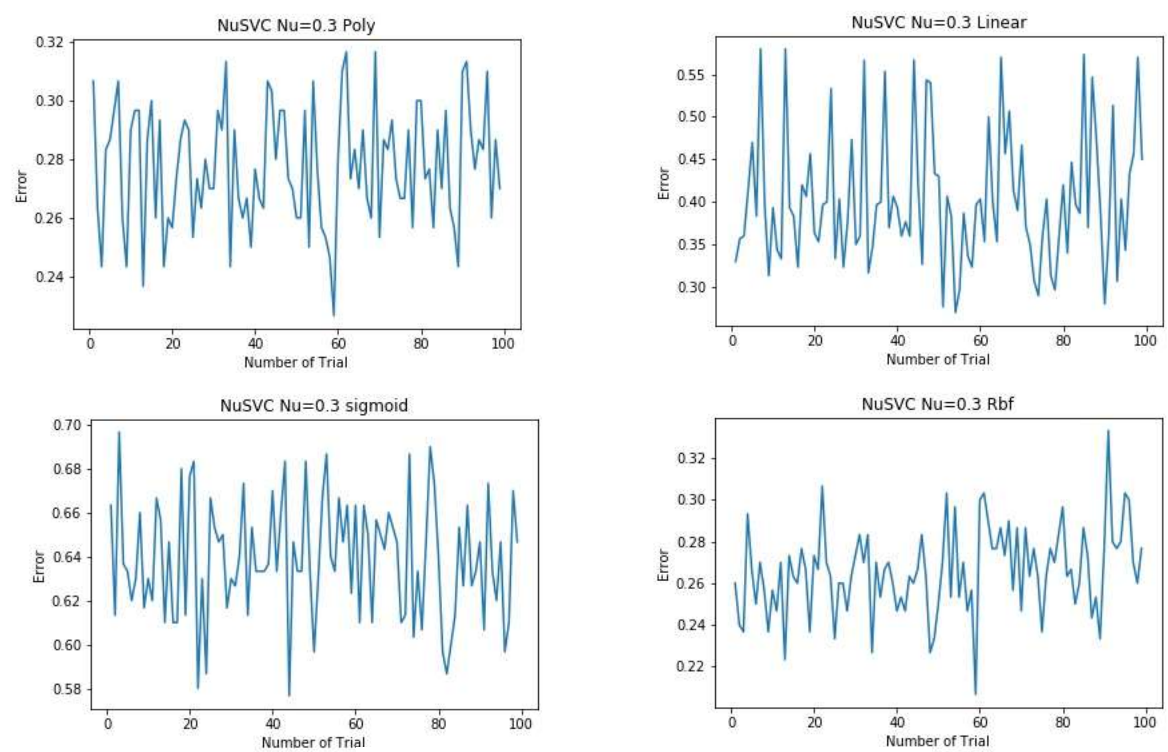

Figure 11:NuSVC Test Set Error

3) Classification Tree

Classification tree pruned and related cost are in Figure 12.

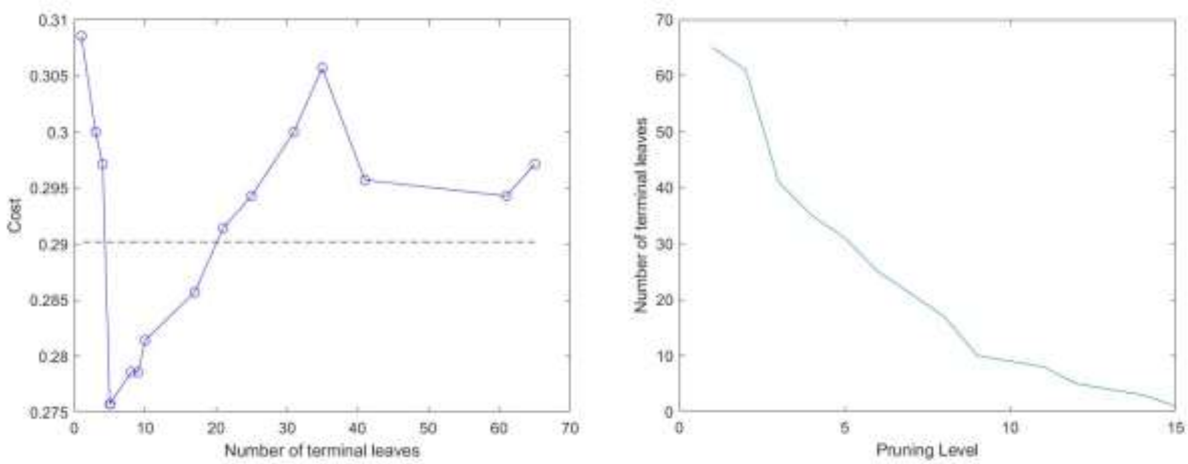

Figure 12: Pruned Level of Main Tree and Cost

Decision tree for classification;

1 if x1<2.5 then node 2 elseif x1>=2.5 then node 3 else 1

2 if x2<0.0268668 then node 4 elseif x2>=0.0268668 then node 5 else 1

3 if x14<2.5 then node 6 elseif x14>=2.5 then node 7 else 1

4 if x3<1.5 then node 8 elseif x3>=1.5 then node 9 else 1

5 if x6<3.5 then node 10 elseif x6>=3.5 then node 11 else 0

6 if x16<0.0311891 then node 12 elseif x16>=0.0311891 then node 13 else 1

7 if x5<0.0695727 then node 14 elseif x5>=0.0695727 then node 15 else 1

8 if x4<7 then node 16 elseif x4>=7 then node 17 else 0

9 if x5<0.0548405 then node 18 elseif x5>=0.0548405 then node 19 else 1

10 if x13<0.0525501 then node 20 elseif x13>=0.0525501 then node 21 else 0

11 if x5<0.0132206 then node 22 elseif x5>=0.0132206 then node 23 else 1

12 if x5<0.00787304 then node 24 elseif x5>=0.00787304 then node 25 else 1

13 if x13<0.0334795 then node 26 elseif x13>=0.0334795 then node 27 else 1

14 if x3<3.5 then node 28 elseif x3>=3.5 then node 29 else 1

15 class = 0

16 if x2<0.0249009 then node 30 elseif x2>=0.0249009 then node 31 else 0

17 class = 1

18 if x5<0.0100509 then node 32 elseif x5>=0.0100509 then node 33 else 1

19 class = 0

20 if x15<1.5 then node 34 elseif x15>=1.5 then node 35 else 0

21 class = 1

22 class = 0

23 if x1<1.5 then node 36 elseif x1>=1.5 then node 37 else 1

24 class = 0

25 if x7<1.5 then node 38 elseif x7>=1.5 then node 39 else 1

26 if x4<0.5 then node 40 elseif x4>=0.5 then node 41 else 0

27 class = 1

28 if x3<2.5 then node 42 elseif x3>=2.5 then node 43 else 1

29 class = 1

30 class = 0

31 class = 1

32 if x12<2.5 then node 44 elseif x12>=2.5 then node 45 else 1

33 if x1<1.5 then node 46 elseif x1>=1.5 then node 47 else 1

34 if x3<2.5 then node 48 elseif x3>=2.5 then node 49 else 0

35 if x14<2.5 then node 50 elseif x14>=2.5 then node 51 else 0

36 if x2<0.043249 then node 52 elseif x2>=0.043249 then node 53 else 1

37 class = 1

38 class = 0

39 class = 1

40 class = 0

41 class = 1

42 if x17<2.5 then node 54 elseif x17>=2.5 then node 55 else 1

43 if x8<0.0348453 then node 56 elseif x8>=0.0348453 then node 57 else 1

44 if x7<1.5 then node 58 elseif x7>=1.5 then node 59 else 1

45 if x2<0.0150716 then node 60 elseif x2>=0.0150716 then node 61 else 0

46 if x2<0.0203139 then node 62 elseif x2>=0.0203139 then node 63 else 1

47 if x5<0.0143442 then node 64 elseif x5>=0.0143442 then node 65 else 1

48 if x20<1.5 then node 66 elseif x20>=1.5 then node 67 else 0

49 class = 1

50 if x13<0.0190706 then node 68 elseif x13>=0.0190706 then node 69 else 0

51 if x13<0.0216134 then node 70 elseif x13>=0.0216134 then node 71 else 0

52 class = 1

53 class = 0

54 if x5<0.0112697 then node 72 elseif x5>=0.0112697 then node 73 else 1

55 if x13<0.0165279 then node 74 elseif x13>=0.0165279 then node 75 else 1

56 class = 1

57 if x6<1.5 then node 76 elseif x6>=1.5 then node 77 else 0

58 class = 0

59 if x17<2.5 then node 78 elseif x17>=2.5 then node 79 else 1

60 class = 1

61 if x13<0.0461933 then node 80 elseif x13>=0.0461933 then node 81 else 0

62 if x19<1.5 then node 82 elseif x19>=1.5 then node 83 else 1

63 if x6<1.5 then node 84 elseif x6>=1.5 then node 85 else 0

64 if x5<0.0140587 then node 86 elseif x5>=0.0140587 then node 87 else 1

65 class = 1

66 class = 0

67 class = 1

68 class = 1

69 class = 0

70 if x13<0.0177992 then node 88 elseif x13>=0.0177992 then node 89 else 0

71 if x13<0.0411078 then node 90 elseif x13>=0.0411078 then node 91 else 1

72 class = 0

73 class = 1

74 class = 0

75 if x5<0.00348083 then node 92 elseif x5>=0.00348083 then node 93 else 1

76 class = 0

77 class = 1

78 class = 1

79 if x13<0.030513 then node 94 elseif x13>=0.030513 then node 95 else 1

80 class = 0

81 class = 1

82 if x13<0.0474646 then node 96 elseif x13>=0.0474646 then node 97 else 1

83 class = 0

84 if x5<0.0301526 then node 98 elseif x5>=0.0301526 then node 99 else 0

85 class = 1

86 class = 1

87 class = 0

88 class = 1

89 if x20<1.5 then node 100 elseif x20>=1.5 then node 101 else 0

90 if x9<2.5 then node 102 elseif x9>=2.5 then node 103 else 1

91 if x2<0.0334197 then node 104 elseif x2>=0.0334197 then node 105 else 0

92 class = 0

93 if x5<0.0264118 then node 106 elseif x5>=0.0264118 then node 107 else 1

94 if x13<0.0250037 then node 108 elseif x13>=0.0250037 then node 109 else 1

95 if x7<2.5 then node 110 elseif x7>=2.5 then node 111 else 1

96 class = 1

97 class = 0

98 if x5<0.0137696 then node 112 elseif x5>=0.0137696 then node 113 else 0

99 class = 1

100 class = 0

101 class = 1

102 if x5<0.0280406 then node 114 elseif x5>=0.0280406 then node 115 else 0

103 if x5<0.0274367 then node 116 elseif x5>=0.0274367 then node 117 else 1

104 class = 1

105 class = 0

106 class = 1

107 if x2<0.0117952 then node 118 elseif x2>=0.0117952 then node 119 else 1

108 if x5<0.00966287 then node 120 elseif x5>=0.00966287 then node 121 else 1

109 class = 0

110 class = 0

111 class = 1

112 class = 1

113 class = 0

114 class = 0

115 class = 1

116 class = 1

117 if x4<0.5 then node 122 elseif x4>=0.5 then node 123 else 0

118 class = 0

119 if x5<0.0270048 then node 124 elseif x5>=0.0270048 then node 125 else 1

120 if x5<0.00662126 then node 126 elseif x5>=0.00662126 then node 127 else 1

121 class = 0

122 class = 0

123 if x3<2.5 then node 128 elseif x3>=2.5 then node 129 else 1

124 class = 0

125 class = 1

126 class = 0

127 class = 1

128 class = 0

129 class = 1