

Group 16

CO 544 Project Report

Milestone 2 (Final Predications to Kaggle)

We used two methods for predictions,

- (i) WEKA
- (ii) Python

Basic Procedure

1)Used weka to build the model using train data and predict test data .

For validation we used 10 fold cross validation. Which can divide the particular data set to 10 sets and validate .

(Train using data set always showed more accuracy than the validation, because train set uses all data to build the model.)

2)Uploaded predicted test result to kaggle

3)Choose the best algorithm from the accuracy

4)Improved (tuned the) choosen algorithms further using python because is more flexible than weka.

Method 1(Using WEKA)

Why WEKA?

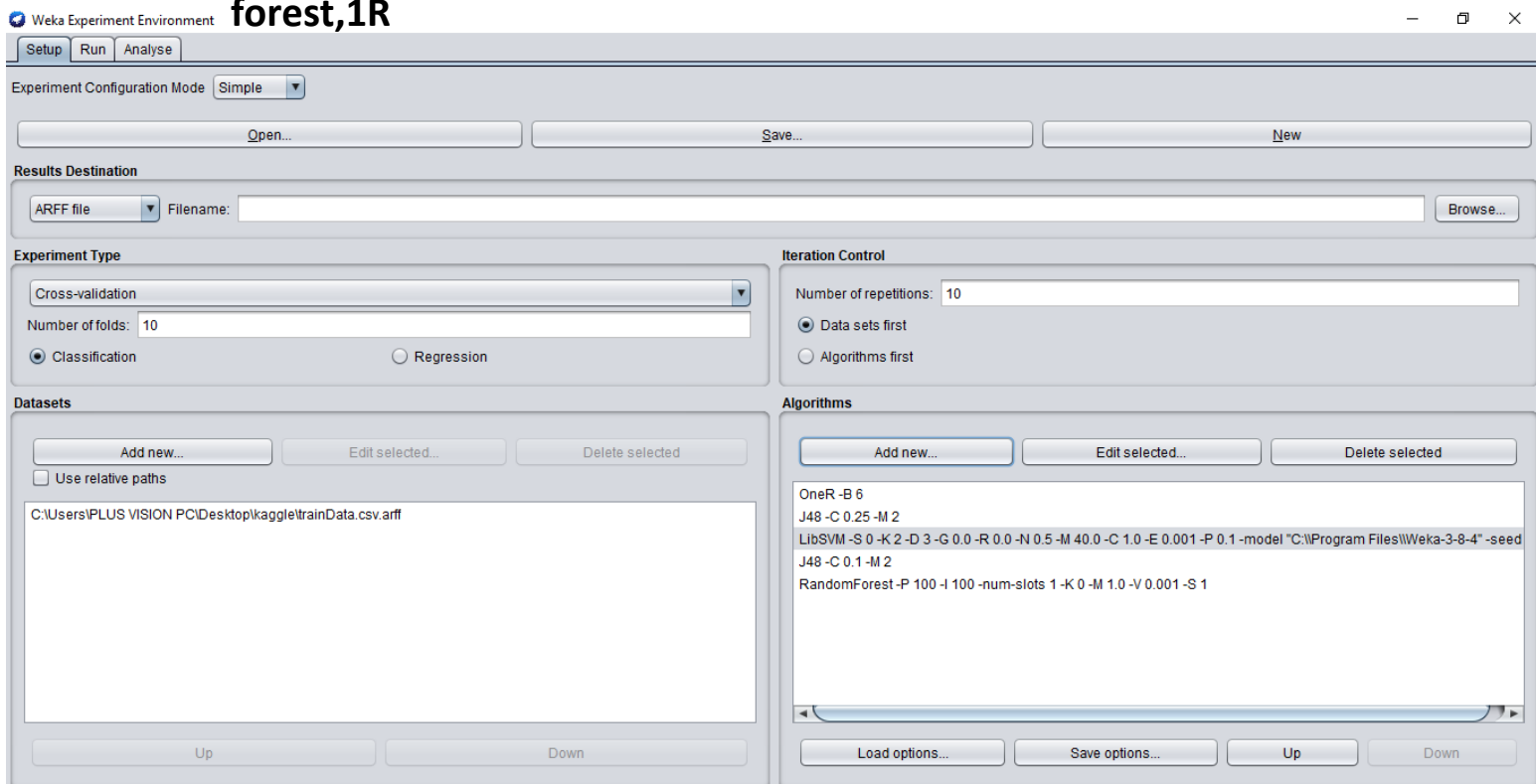
Weka gives a GUI which made comparisons easy for the training data.

We tried various algorithms supervised classification algorithms on the training data. We can get a percentage of correctly classified instances in Weka GUI.

- K-nearest neighbors
- Support Vector Machines
- Decision Trees
- Random Forest

How to compare the algorithms?

1. Weka -> Experimenter
2. Choose the train data set
3. Choose 10 folds cross validation. Repetition =10
4. Algorithms- J48(Confidence factor=0.1 and 0.25),SVM, Random forest,1R



5. Analyse using “Paired T-tests”

Weka Experiment Environment

Setup Run Analyse

Source

Got 500 results

File... Database... Experiment

Actions

Perform test Save output Open Explorer...

Configure test

Testing with: Paired T-Tester (corrected)

Select rows and cols: Rows Cols Swap

Comparison field: Percent_correct

Significance: 0.05

Sorting (asc.) by: <default>

Test base: Select

Displayed Columns: Select

Show std. deviations: ☐

Output Format: Select

Test output

Tester: weka.experiment.PairedCorrectedTTester -G 4,5,6 -D 1 -R 2 -S 0.05 -result-matrix "weka.experiment.ResultMatrixPlainText -me

Analysing: Percent_correct

Datasets: 1

Resultsets: 5

Confidence: 0.05 (two tailed)

Sorted by: -

Date: 5/10/20, 4:14 PM

Dataset	(1) rules.On	(2) trees	(3) funct	(4) trees	(5) trees
trainData	(100) 83.88	85.40	62.28 *	85.34	86.81
	(v/ /*)	(0/1/0)	(0/0/1)	(0/1/0)	(0/1/0)

Key:

(1) rules.OneR '-B 6' -3459427003147861443

(2) trees.J48 '-C 0.25 -M 2' -217733168393644444

(3) functions.LibSVM '-S 0 -K 2 -D 3 -G 0.0 -R 0.0 -N 0.5 -M 40.0 -C 1.0 -E 0.001 -P 0.1 -model \\C:\\\\Program Files\\\\Weka-3-8-4\\'

(4) trees.J48 '-C 0.1 -M 2' -217733168393644444

(5) trees.RandomForest '-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1' 1116839470751428698

Result list

16:14:07 - Available resultsets

16:14:14 - Percent_correct - rules.OneR '-B 6' -34594270031-

Dataset	(1) rules.On	(2) trees	(3) funct	(4) trees	(5) trees
trainData	(100) 83.88	85.40	62.28 *	85.34	86.81
	(v/ /*)	(0/1/0)	(0/0/1)	(0/1/0)	(0/1/0)

Key:

- (1) rules.OneR '-B 6' -3459427003147861443
- (2) trees.J48 '-C 0.25 -M 2' -217733168393644444
- (3) functions.LibSVM '-S 0 -K 2 -D 3 -G 0.0 -R 0.0 -N 0.5 -M 40.0 -C 1.0 -E 0.001 -P 0.1 -model \\C:\\\\Program Files\\\\Weka-
- (4) trees.J48 '-C 0.1 -M 2' -217733168393644444
- (5) trees.RandomForest '-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1' 1116839470751428698

6. From the above results it is clearly concluded that Random Forest and J48 tree gives the best accuracy. So they were used to build model in “weka explorer”.

(i)J48(C4.5)

Confidence factor =0.25

1)train set

Test options

- ☒ Use training set
- ☐ Supplied test set
- ☐ Cross-validation Folds
- ☐ Percentage split %
-

(Nom) A16

Result list (right-click for options)

- 13:08:05 - rules.ZeroR
- 13:08:22 - trees.J48

Classifier output

```
=== Evaluation on training set ===

Time taken to test model on training data: 0.02 seconds

=== Summary ===

Correctly Classified Instances      501      90.7609 %
Incorrectly Classified Instances    51      9.2391 %
Kappa statistic                    0.8147
Mean absolute error                 0.1629
Root mean squared error            0.2846
Relative absolute error            32.7754 %
Root relative squared error       57.0763 %
Total Number of Instances         552

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall  F-Measure  MCC   ROC Area  PRC Area  Class
          0.922    0.104    0.883     0.922    0.902     0.815  0.929    0.888    Success
          0.896    0.078    0.930     0.896    0.913     0.815  0.929    0.914    Failure
Weighted Avg.   0.908    0.090    0.909     0.908    0.908     0.815  0.929    0.902
```

2) cross validate

Classifier

J48 - C 0.25 - M 2

Test options

- ☐ Use training set
- ☐ Supplied test set
- ☒ Cross-validation Folds
- ☐ Percentage split %
-

(Nom) A16

Result list (right-click for options)

- 16:20:12 - trees.J48
- 16:20:16 - trees.J48

Classifier output

```
Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      473      85.6884 %
Incorrectly Classified Instances    79      14.3116 %
Kappa statistic                    0.7125
Mean absolute error                 0.2005
Root mean squared error            0.3427
Relative absolute error            40.3261 %
Root relative squared error       68.7375 %
Total Number of Instances         552

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall  F-Measure  MCC   ROC Area  PRC Area  Class
          0.855    0.141    0.838     0.855    0.847     0.713  0.875    0.815    Success
          0.859    0.145    0.873     0.859    0.866     0.713  0.875    0.857    Failure
Weighted Avg.   0.857    0.143    0.857     0.857    0.857     0.713  0.875    0.838
```

Remarks: Even though Cross validation shows less accuracy than test the tree was same because at the end cross validation too uses all data points to build up the model

C4.5 (**J48**) is an **algorithm** used to generate a **decision tree** developed by Ross Quinlan. Extension of ID3

Confidence Factor=0.5

1) Train set

Choose J48 -C 0.5 -M 2

Test options

☒ Use training set
☐ Supplied test set Set...
☐ Cross-validation Folds 10
☐ Percentage split % 66
More options...

(Nom) A16

Start Stop

Result list (right-click for options)

- 13:08:05 - rules.ZeroR
- 13:08:22 - trees.J48
- 13:40:49 - trees.J48

Classifier output

--- Evaluation on training set ---

Time taken to test model on training data: 0.01 seconds

=== Summary ===

Correctly Classified Instances	516	93.4783 %
Incorrectly Classified Instances	36	6.5217 %
Kappa statistic	0.8688	
Mean absolute error	0.1117	
Root mean squared error	0.2353	
Relative absolute error	22.474 %	
Root relative squared error	47.1969 %	
Total Number of Instances	552	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.929	0.061	0.929	0.929	0.929	0.869	0.966	0.957	Success
	0.939	0.071	0.939	0.939	0.939	0.869	0.966	0.957	Failure
Weighted Avg.	0.935	0.066	0.935	0.935	0.935	0.869	0.966	0.957	

=== Confusion Matrix ===

a	b	<-- classified as
237	18	a = Success
18	279	b = Failure

Remarks:Increasing the confidence factor increased the accuracy.

2)Cross validation

☐ Use training set
☐ Supplied test set Set...
☒ Cross-validation Folds 10
☐ Percentage split % 66
More options...

(Nom) A16

Start Stop

Result list (right-click for options)

- 16:20:16 - trees.J48
- 16:25:33 - trees.J48
- 16:25:36 - trees.J48

Classifier output

Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	468	84.7826 %
Incorrectly Classified Instances	84	15.2174 %
Kappa statistic	0.6942	
Mean absolute error	0.1947	
Root mean squared error	0.3572	
Relative absolute error	39.1727 %	
Root relative squared error	71.6463 %	
Total Number of Instances	552	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.843	0.148	0.830	0.843	0.837	0.694	0.868	0.799	Success
	0.852	0.157	0.863	0.852	0.858	0.694	0.868	0.849	Failure
Weighted Avg.	0.848	0.153	0.848	0.848	0.848	0.694	0.868	0.826	

=== Confusion Matrix ===

a	b	<-- classified as
215	40	a = Success
44	253	b = Failure

(ii) Random forest

RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1

ng set

test set

Set...

Validation Folds 10

Split % 66

More options...

Stop

(click for options)

es ZeroR

es J48

es J48

es RandomForest

es RandomForest

Classifier output

--- Evaluation on training set ---

Time taken to test model on training data: 0.05 seconds

=== Summary ===

Correctly Classified Instances	552	100	%
Incorrectly Classified Instances	0	0	%
Kappa statistic	1		
Mean absolute error	0.0876		
Root mean squared error	0.1262		
Relative absolute error	17.6172	%	
Root relative squared error	25.3046	%	
Total Number of Instances	552		

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	Success
	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	Failure
Weighted Avg.	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	

Remarks:Very high accuracy. BagsizePercentage and batch size and nmber of slots were changed to get the highest accuracy with lowest mean absolute error.

Step:

By uploading the results of the above algorithms to the kaggle we got the above decision tree and the Random forest algorithm gave the the best accuracy for test data set (in the public leader board).

Therefore we choose J48 classifier and Random Forest classification with python for further development.

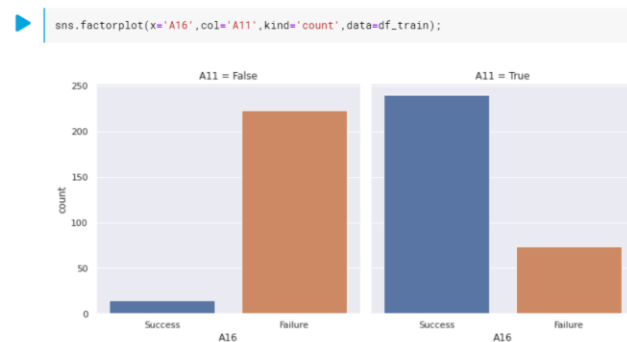
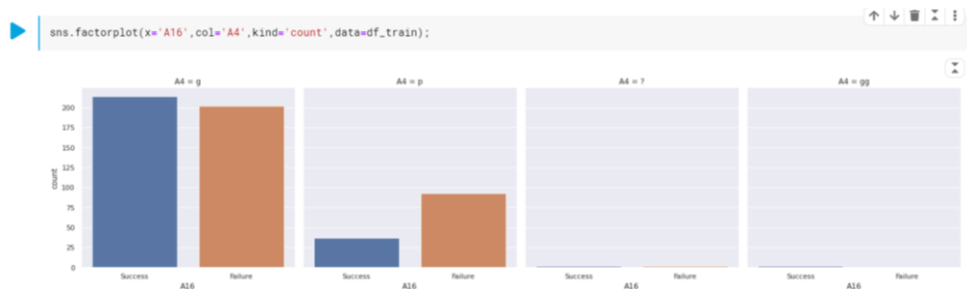
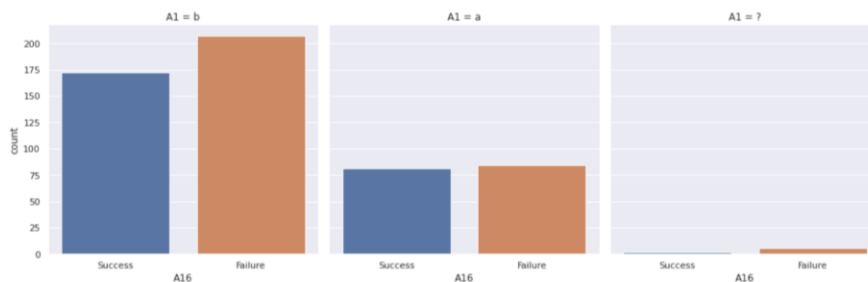
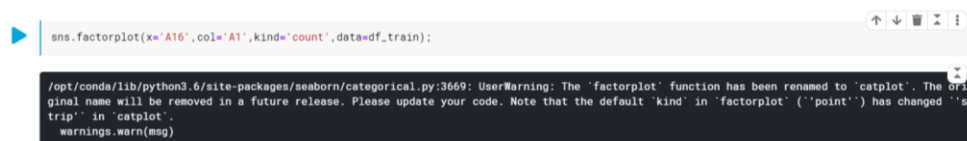
Python

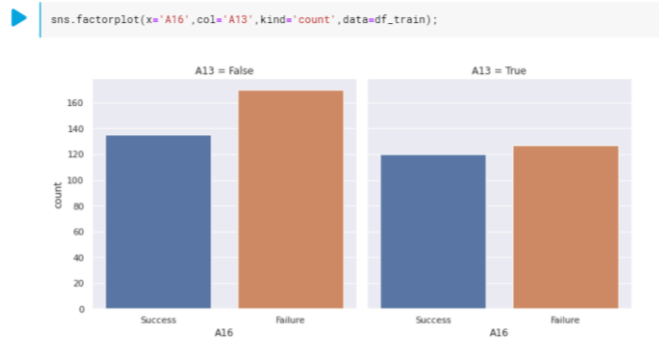
Why python?

Even though weka gives GUI and a great environment for predictions, it is less flexible and so as we explore we got to know that the Python in ML gives more degree of freedom when exploring and tune and tweeking the algorithms.

Following is the python procedure

Explore the features





Target values



Check missing values in training and test datasets.

```
#replace ? with nan
data=data.replace("?",np.NaN)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 690 entries, 0 to 137
Data columns (total 15 columns):
A1      678 non-null object
A2      678 non-null object
A3      684 non-null object
A4      684 non-null object
A5      690 non-null float64
A6      681 non-null object
A7      690 non-null int64
A8      690 non-null bool
A9      681 non-null object
A10     690 non-null float64
A11     690 non-null bool
A12     690 non-null int64
A13     690 non-null bool
A14     677 non-null object
A15     690 non-null object
dtypes: bool(3), float64(2), int64(2), object(8)
memory usage: 72.1+ KB
```


Process Missing values

```
#numerical imputer
imp = SimpleImputer(missing_values=np.nan, strategy='mean');
data[num_cols] = imp.fit_transform(data[num_cols])
#categorical imputer
cat_imp = SimpleImputer(missing_values=np.nan, strategy='most_frequent');
data[cat_cols] = cat_imp.fit_transform(data[cat_cols])
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 698 entries, 0 to 137
Data columns (total 15 columns):
 A1      698 non-null object
 A2      698 non-null float64
 A3      698 non-null object
 A4      698 non-null object
 A5      698 non-null float64
 A6      698 non-null object
 A7      698 non-null float64
 A8      698 non-null bool
 A9      698 non-null object
 A10     698 non-null float64
 A11     698 non-null bool
 A12     698 non-null float64
 A13     698 non-null bool
 A14     698 non-null float64
 A15     698 non-null object
dtypes: bool(3), float64(6), object(6)
```

Encode nominal attributes with numbers

```
#Get dummies
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
data[cat_cols] = data[cat_cols].apply(lambda col: le.fit_transform(col))
data[cat_cols].head(10)
```

```
0):
```

	A1	A3	A4	A6	A8	A9	A11	A13	A15
0	1	1	0	12	1	7	1	0	0
1	0	1	0	10	1	3	1	0	0
2	0	1	0	10	0	3	1	0	0
3	1	1	0	12	1	7	1	1	0
4	1	1	0	1	1	7	1	0	0
5	1	1	0	1	0	7	1	0	2
6	0	1	0	1	1	7	1	1	0
7	0	1	0	13	1	3	1	0	0
8	0	1	0	10	1	7	1	1	0
9	0	1	0	1	1	3	1	0	0

Selecting Hyper parameters for models

Split the train data set

```
[15]: from sklearn.model_selection import train_test_split
      X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

Decision tree

```
#Decision Tree
from sklearn import tree
for depth in range(1,15):
    clf=tree.DecisionTreeClassifier(max_depth=depth);
    clf.fit(X_train,y_train)
    print("depth=%d score_train=%f score_test=%f"%(depth,clf.score(X_train,y_train),clf.score(X_test,y_test)))
```

```
depth=1 score_train=0.840580 score_test=0.833333
depth=2 score_train=0.842995 score_test=0.840580
depth=3 score_train=0.850242 score_test=0.840580
depth=4 score_train=0.886473 score_test=0.876812
depth=5 score_train=0.910628 score_test=0.855072
depth=6 score_train=0.946860 score_test=0.855072
depth=7 score_train=0.973430 score_test=0.847826
depth=8 score_train=0.985587 score_test=0.855072
depth=9 score_train=0.992754 score_test=0.862319
depth=10 score_train=0.997585 score_test=0.833333
depth=11 score_train=1.000000 score_test=0.811594
depth=12 score_train=1.000000 score_test=0.818841
depth=13 score_train=1.000000 score_test=0.862319
depth=14 score_train=1.000000 score_test=0.840580
```

Random Forest

```
#Random Forest
from sklearn.ensemble import RandomForestClassifier
randomForestClassifier=RandomForestClassifier(n_estimators= 1000, random_state=42)
randomForestClassifier.fit(X_train,y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                        criterion='gini', max_depth=None, max_features='auto',
                        max_leaf_nodes=None, max_samples=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, n_estimators=1000,
                        n_jobs=None, oob_score=False, random_state=42, verbose=0,
                        warm_start=False)
```

```
randomForestClassifier.score(X_test,y_test)
```

```
0.9857971814492754
```

Considering the each model performance after selecting the hyper parameters , Select the final model and train the model for the whole train data set.

Brief Procedure of how weka used to predict

Milestone 1(Before kaggle introduced)

Prediction margin-

defined as the difference between the probability predicted for the actual class and the highest probability predicted for the other classes. A margin of 1 means that the correct class is predicted with 100% confidence (very good), a margin of -1 means that an incorrect class is predicted with 100% confidence (very bad).

Procedure

1).Both Training and Test data sets were preprocessed and made to ARFF formats. For test data sets Attribute names were given and the Class (Which is to be predicted is names as "A16")

[illegible]

Figure 1: Modified Test data set

```

C:\Users\PLUS VISION PC\Desktop\Semester 6\CO 544 - Machine Learning and Data Mining\Project\2\testdata_10%.arff - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
config.js testresult1 data.csv.arff testdata_10%.arff final1.arff final2.arff final1aa.arff
1 @relation 'testdata_10%\%-weka.filters.unsupervised.attribute.StringToNominal-R1ast'
2
3 @attribute A1 {b,a}
4 @attribute A2 numeric
5 @attribute A3 {u,y,l}
6 @attribute A4 {g,p,gg}
7 @attribute A5 numeric
8 @attribute A6 {w,q,c,x,i,d,e,aa,cc,ff,m,k,j,r}
9 @attribute A7 numeric
10 @attribute A8 {TRUE,FALSE}
11 @attribute A9 {v,h,bb,ff,j,z,o,dd,n}
12 @attribute A10 numeric
13 @attribute A11 {TRUE,FALSE}
14 @attribute A12 numeric
15 @attribute A13 {FALSE,TRUE}
16 @attribute A14 numeric
17 @attribute A15 {g,s,p}
18 @attribute A16 {Success,Failure}
19
20 @data
21 b,32.67,y,p,9,w,0,FALSE,h,5.25,TRUE,0,TRUE,154,g,?
22 ?,20.08,u,g,0.125,q,768,TRUE,v,1,FALSE,1,FALSE,240,g,?
23 b,20.08,u,g,0.25,q,0,FALSE,v,0.125,FALSE,0,FALSE,200,g,?
24 b,22.17,u,g,2.25,i,10,FALSE,v,0.125,FALSE,0,FALSE,160,g,?
25 a,27.25,u,g,0.29,m,108,TRUE,h,0.125,FALSE,1,TRUE,272,g,?
26 b,31.58,y,p,0.75,aa,0,FALSE,v,3.5,FALSE,0,TRUE,320,g,?
27 a,20.83,u,g,8.5,c,351,FALSE,v,0.165,FALSE,0,FALSE,0,g,?
28 b,48.08,u,g,3.75,i,2,FALSE,bb,1,FALSE,0,FALSE,100,g,?
29 b,29.83,u,g,3.5,c,0,FALSE,v,0.165,FALSE,0,FALSE,216,g,?
30 a,41.58,u,g,1.04,aa,237,FALSE,v,0.665,FALSE,0,FALSE,240,g,?
31 b,33.17,u,g,1.04,r,31285,FALSE,h,6.5,TRUE,0,TRUE,164,g,?
32 a,18.92,u,g,9,aa,591,TRUE,v,0.75,TRUE,2,FALSE,88,g,?
33 a,24.75,u,g,3,q,500,TRUE,h,1.835,TRUE,19,FALSE,0,g,?
34 b,21,y,p,4.79,w,300,TRUE,v,2.25,TRUE,1,TRUE,80,g,?
35
36

```

Figure 2: Modified test data in ARFF format

2) Training Data set classified Using J48 tree(Use as training data set)

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose J48 -C 0.25 -M 2

Test options

☒ Use training set

☐ Supplied test set Set...

☐ Cross-validation Folds 10

☐ Percentage split % 66

More options...

(Nom) A16

Start Stop

Result list (right-click for options)

17:15:49 - trees.J48

Classifier output

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances	501	90.7609 %
Incorrectly Classified Instances	51	9.2391 %
Kappa statistic	0.8147	
Mean absolute error	0.1629	
Root mean squared error	0.2846	
Relative absolute error	32.7754 %	
Root relative squared error	57.0763 %	
Total Number of Instances	552	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.922	0.104	0.883	0.922	0.902	0.815	0.929	0.888	Success
	0.896	0.078	0.930	0.896	0.913	0.815	0.929	0.914	Failure
	0.909	0.098	0.908	0.908	0.908	0.815	0.929	0.902	

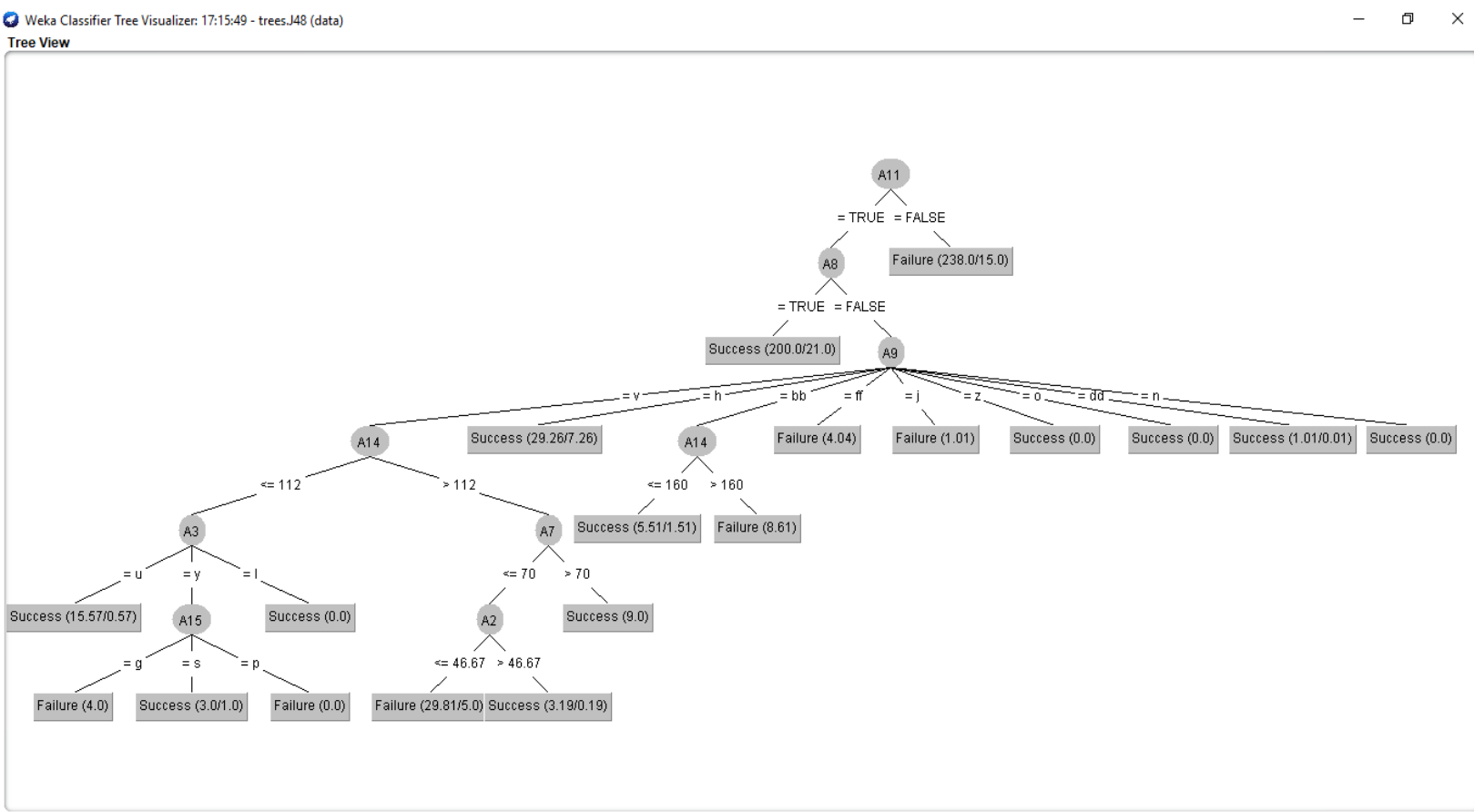
=== Confusion Matrix ===

a b <-- classified as

235	20	a = Success
31	266	b = Failure

Correctly Classified 90.7609%

Visualised tree(for training data set)



3) Supplied the test data set and classified Using J48 algorithm.

Results were saved.

Results

Dear all,

Below are the results. This was an intermediate activity to assist you with the project work, well done on your submissions.

Rank	Group No:	Accuracy
1	10	100.00%
2	2	92.86%
2	3	92.86%
2	5	92.86%
2	16	92.86%
2	17	92.86%
2	20	92.86%
3	4	85.71%

Figure 3: Final Predictions results on Feels

Our predictions were 92.86% accurate ranked 2.

Conclusion

Prediction margin: This is defined as the difference between the probability predicted for the actual class and the highest probability predicted for the other classes. A margin of 1 means that the correct class is predicted with 100% confidence (very good), a margin of -1 means that an incorrect class is predicted with 100% confidence (very bad).