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?

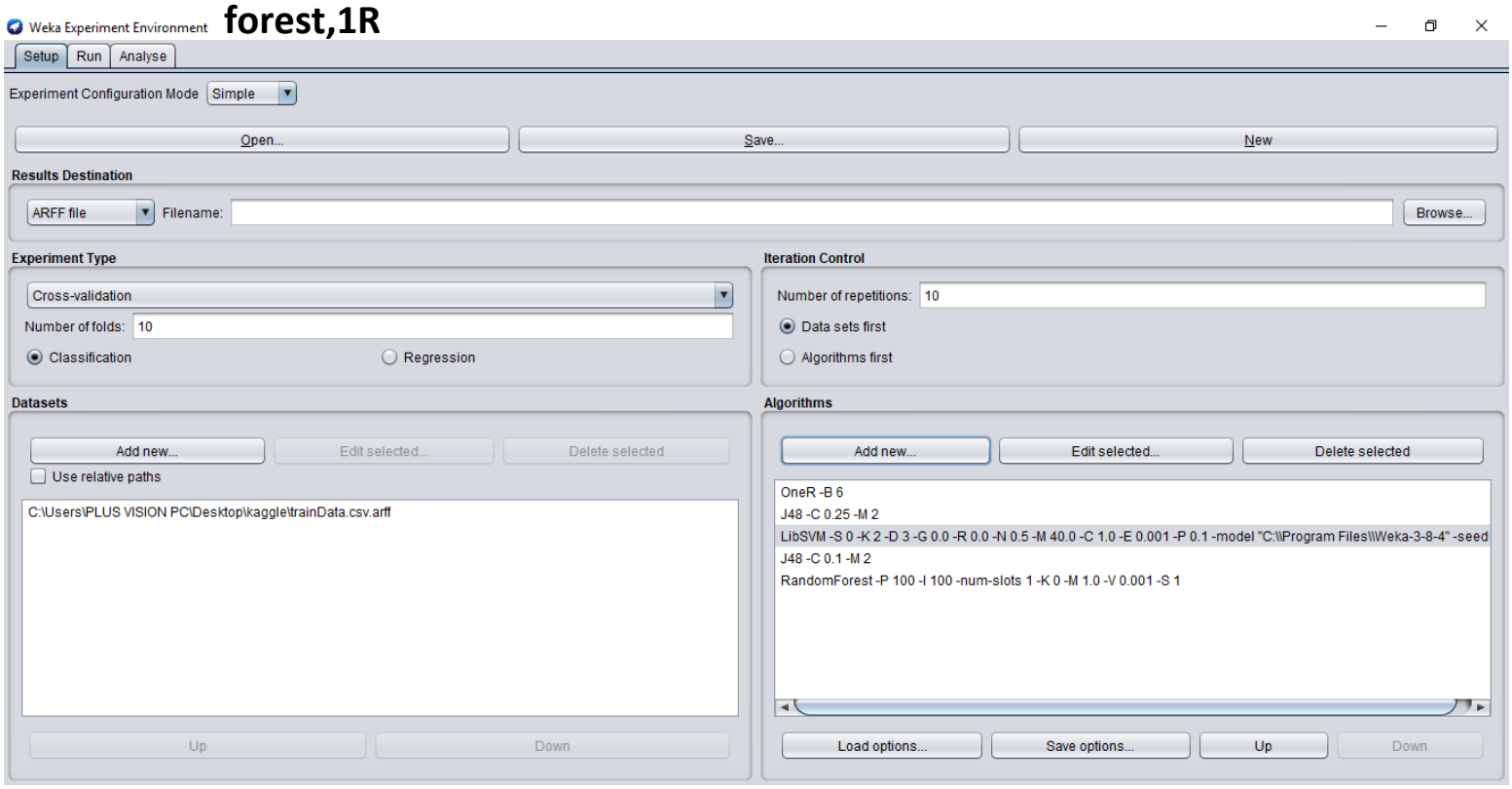
Weka gives a GUI which made comparisions easy for the training data.

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

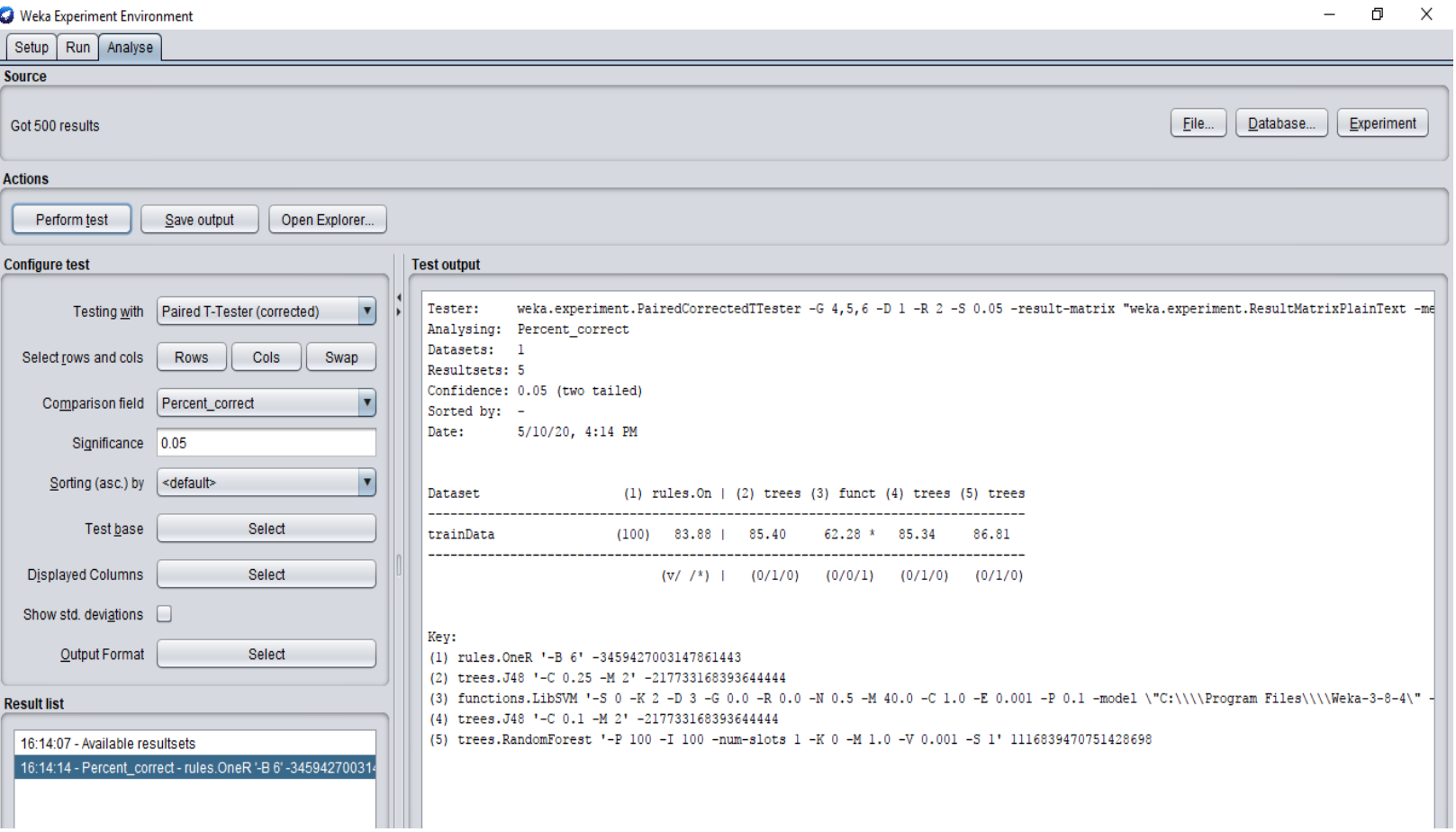
- K-nearest neighbors
- Support Vector Machines
- Decision Trees
- Random Forest

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”



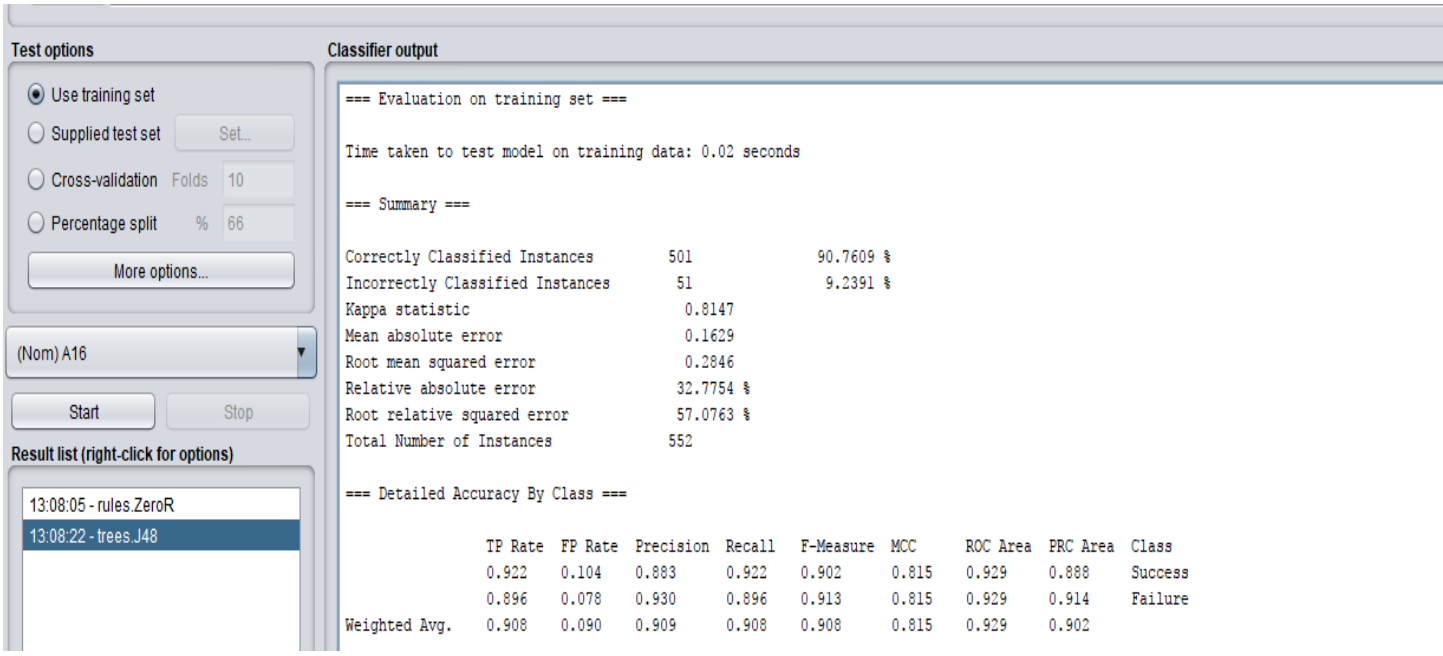

```
Dataset                   (1) rules.On | (2) trees (3) funct (4) trees (5) trees
------------------------------------------------------------------------------------
trainData                (100)   83.88 |   85.40     62.28 *   85.34     86.81
------------------------------------------------------------------------------------
                               (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”.

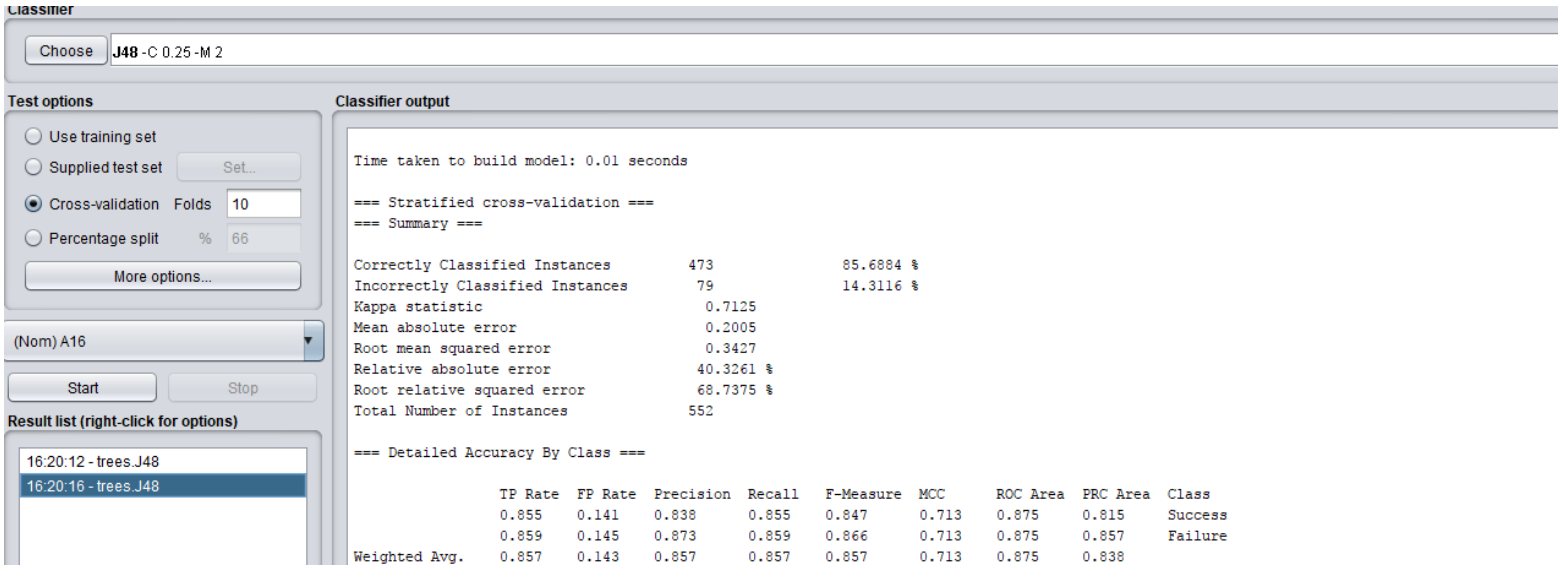
(i)J48(C4.5)

Confidence factor =0.25

1)train set



2) cross validate

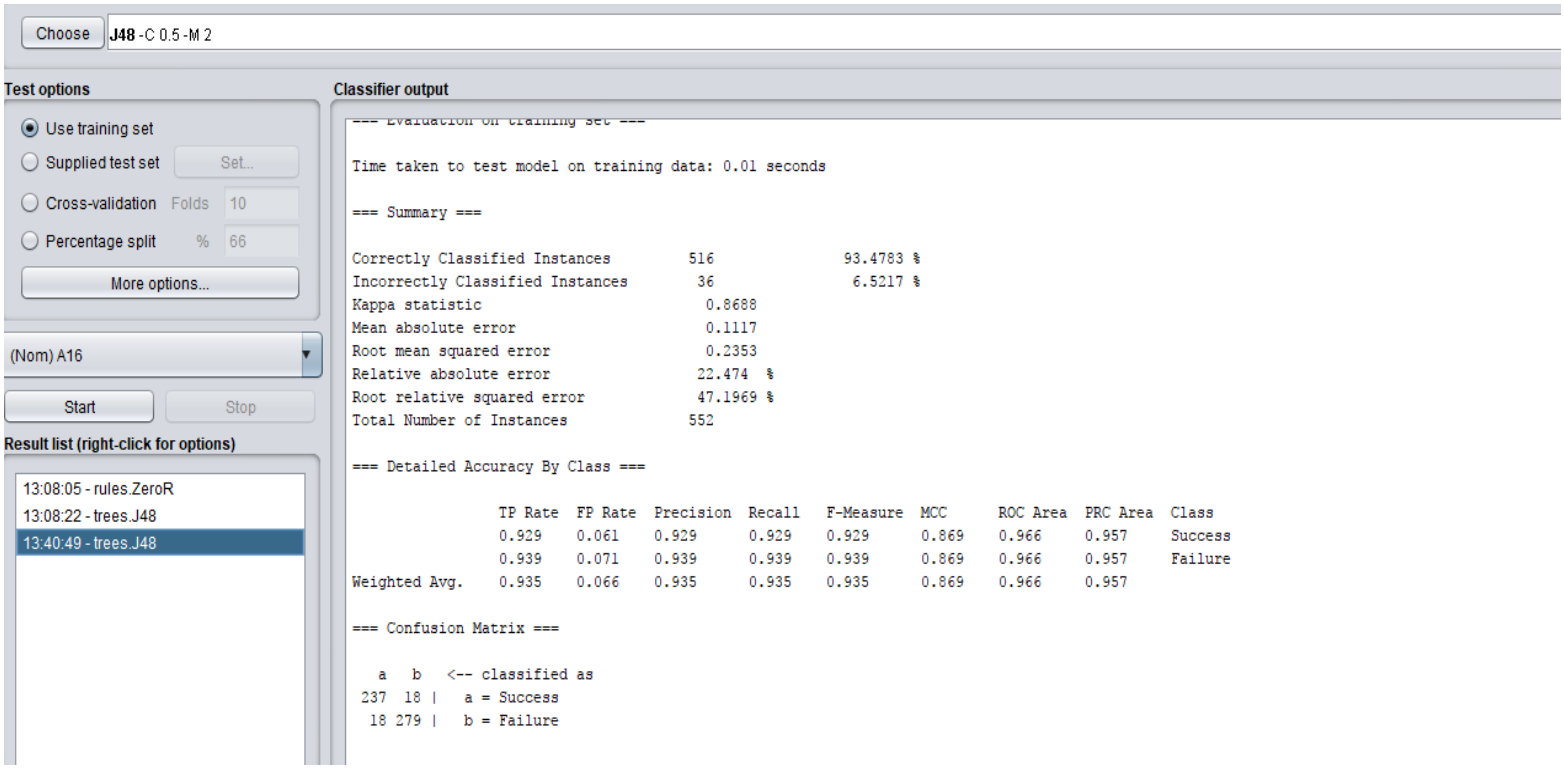


Remarks:Even though Cross validation shows less accuracy than test the tree was same because at the end cross validation too uses all data points to build up the model

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

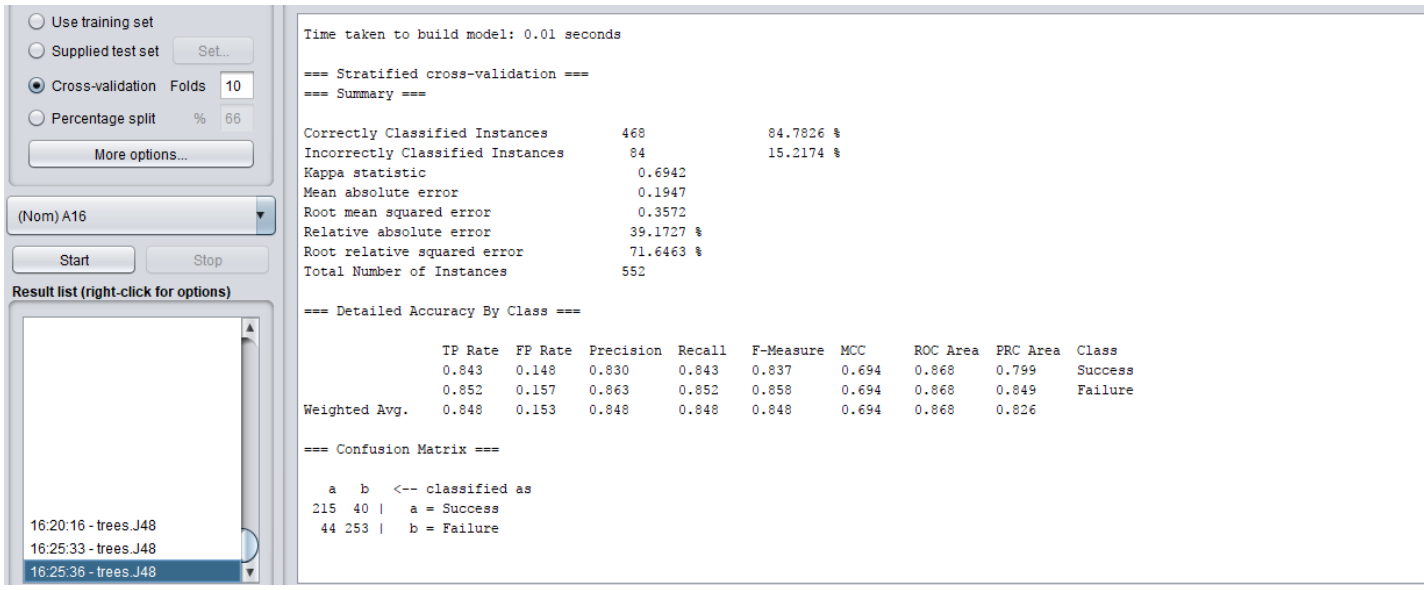
Confidence Factor=0.5

1) Train set

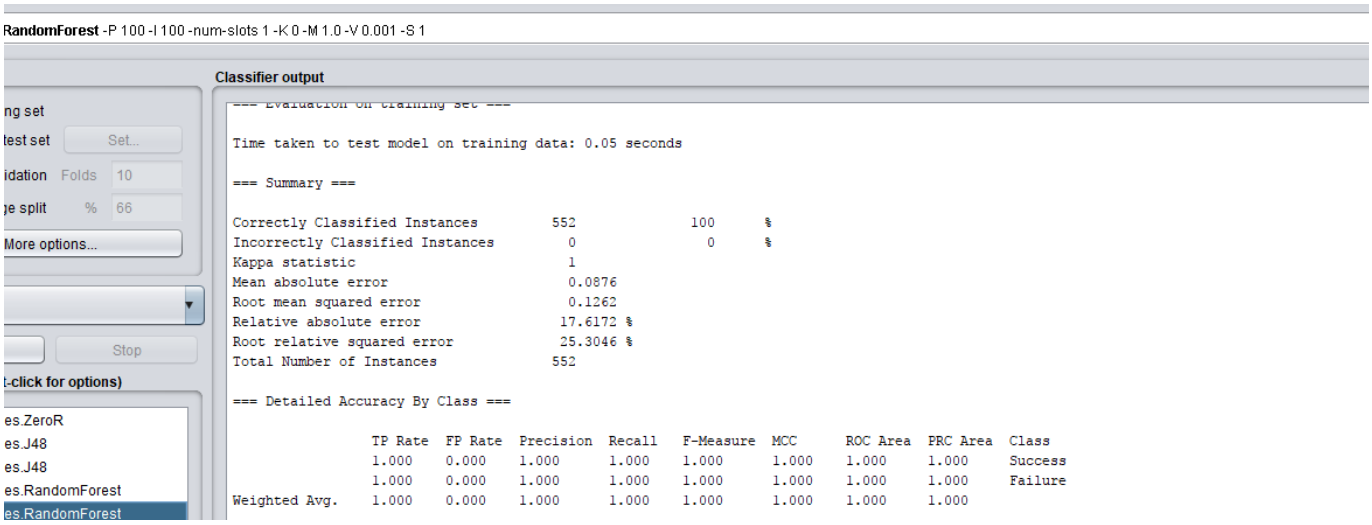


Remarks:Increasing the confidence factor increased the accuracy.

2)Cross validation



(ii) Random forest



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?

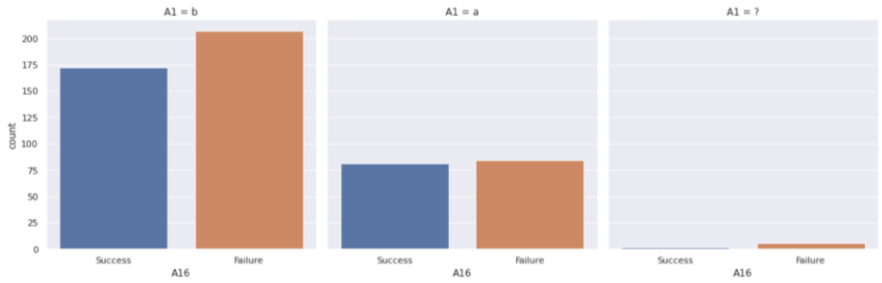
Even though weka gives GUI and a great envioranment for predoctions, it is less flexible and so as we explore we got to know that the Python in ML gives more degree of freedom when exploring and tune and tweeking the algoithms.

Following is the python procedure

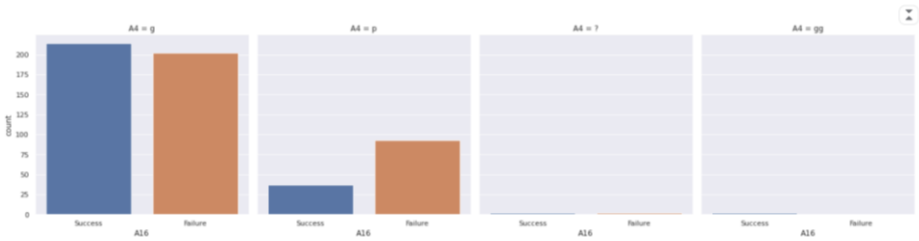
Explore the features

```
sns.factorplot(x='A16',col='A1',kind='count',data=df_train);
```

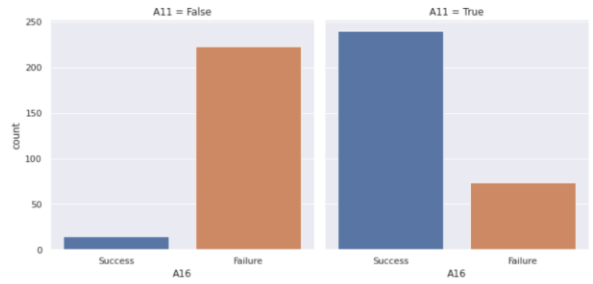
```
/opt/conda/lib/python3.6/site-packages/seaborn/categorical.py:3669: UserWarning: The `factorplot` function has been renamed to `catplot`. The ori
ginal name will be removed in a future release. Please update your code. Note that the default `kind` in `factorplot` (`'point'`) has changed `'s
trip'` in `catplot`.
  warnings.warn(msg)
```



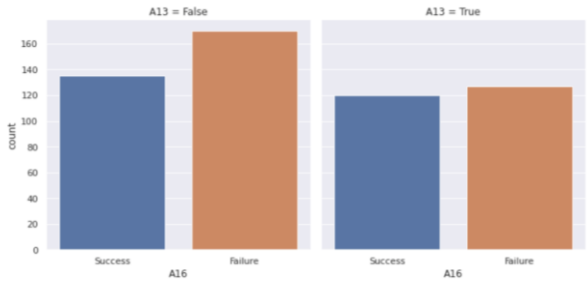

```
sns.factorplot(x='A16',col='A4',kind='count',data=df_train);
```




```
sns.factorplot(x='A16',col='A11',kind='count',data=df_train);
```



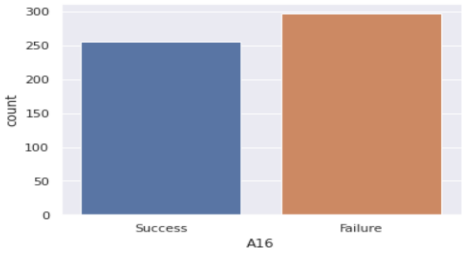
```
sns.factorplot(x='A16',col='A13',kind='count',data=df_train);
```



Target values

```
#target value plot
sns.countplot(x='A16',data=df_train)
```

Out[29]
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe77c5d06a0>
```



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: 690 entries, 0 to 137
Data columns (total 15 columns):
A1      690 non-null object
A2      690 non-null float64
A3      690 non-null object
A4      690 non-null object
A5      690 non-null float64
A6      690 non-null object
A7      690 non-null float64
A8      690 non-null bool
A9      690 non-null object
A10     690 non-null float64
A11     690 non-null bool
A12     690 non-null float64
A13     690 non-null bool
A14     690 non-null float64
A15     690 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)
```

	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

```
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=0.985507 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)
```

```
0.9057971014492754
```

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

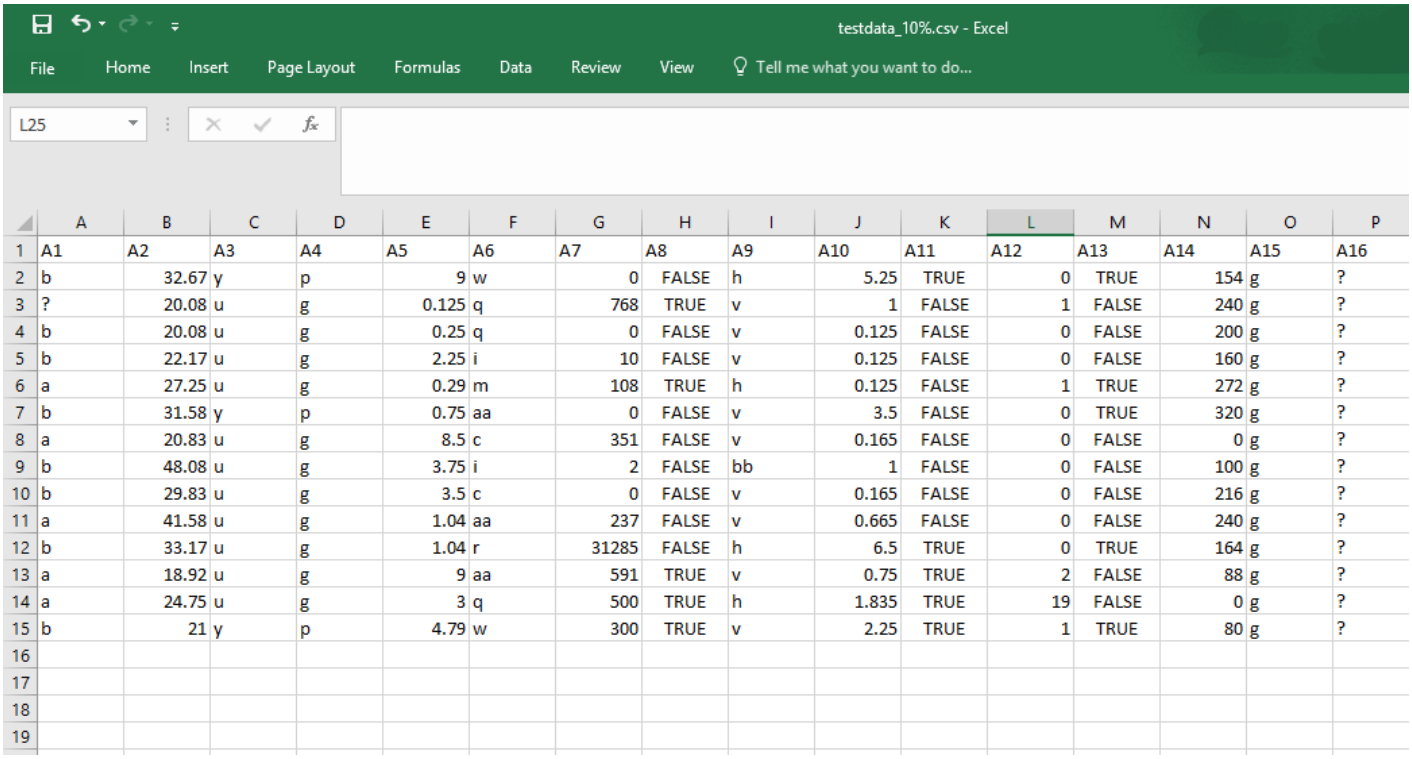
Milestone 1(Before kaggel introduced)

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")



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16
2	b	32.67	y	p	9	w	0	FALSE	h	5.25	TRUE	0	TRUE	154	g	?
3	?	20.08	u	g	0.125	q	768	TRUE	v	1	FALSE	1	FALSE	240	g	?
4	b	20.08	u	g	0.25	q	0	FALSE	v	0.125	FALSE	0	FALSE	200	g	?
5	b	22.17	u	g	2.25	i	10	FALSE	v	0.125	FALSE	0	FALSE	160	g	?
6	a	27.25	u	g	0.29	m	108	TRUE	h	0.125	FALSE	1	TRUE	272	g	?
7	b	31.58	y	p	0.75	aa	0	FALSE	v	3.5	FALSE	0	TRUE	320	g	?
8	a	20.83	u	g	8.5	c	351	FALSE	v	0.165	FALSE	0	FALSE	0	g	?
9	b	48.08	u	g	3.75	i	2	FALSE	bb	1	FALSE	0	FALSE	100	g	?
10	b	29.83	u	g	3.5	c	0	FALSE	v	0.165	FALSE	0	FALSE	216	g	?
11	a	41.58	u	g	1.04	aa	237	FALSE	v	0.665	FALSE	0	FALSE	240	g	?
12	b	33.17	u	g	1.04	r	31285	FALSE	h	6.5	TRUE	0	TRUE	164	g	?
13	a	18.92	u	g	9	aa	591	TRUE	v	0.75	TRUE	2	FALSE	88	g	?
14	a	24.75	u	g	3	q	500	TRUE	h	1.835	TRUE	19	FALSE	0	g	?
15	b	21	y	p	4.79	w	300	TRUE	v	2.25	TRUE	1	TRUE	80	g	?
16																
17																
18																
19																

Figure 1: Modified Test data set

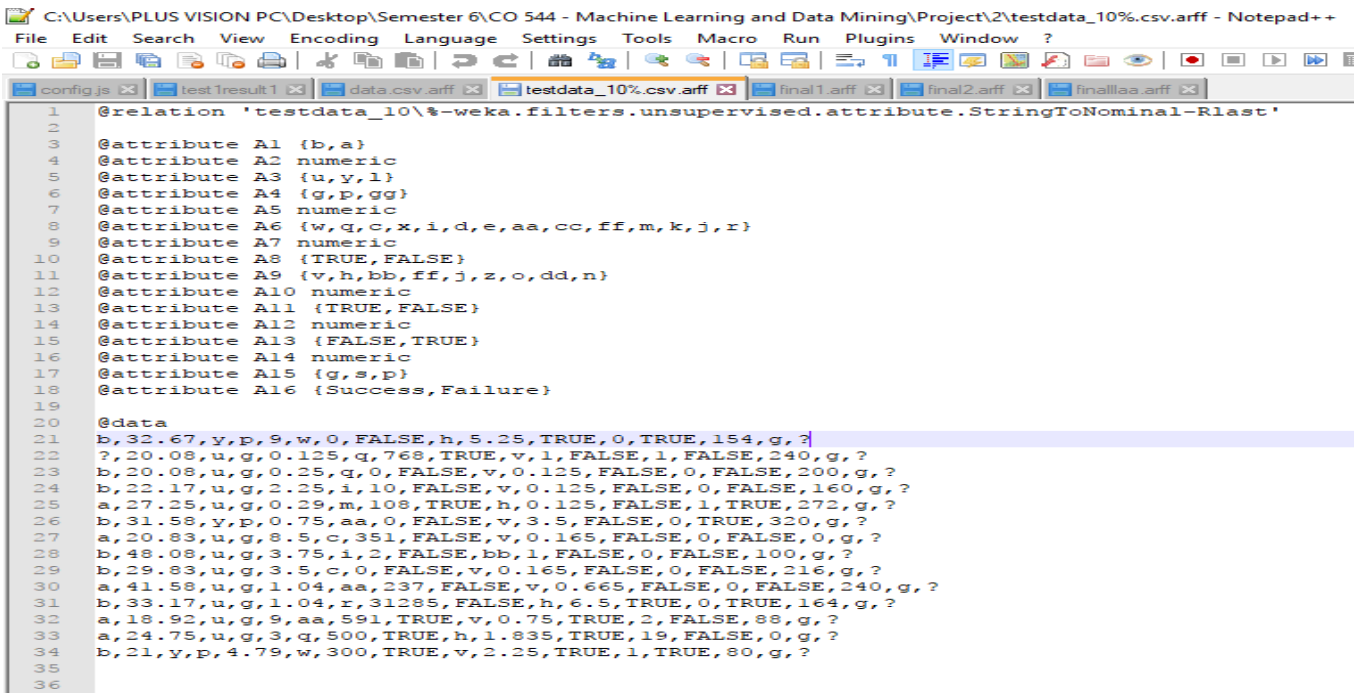
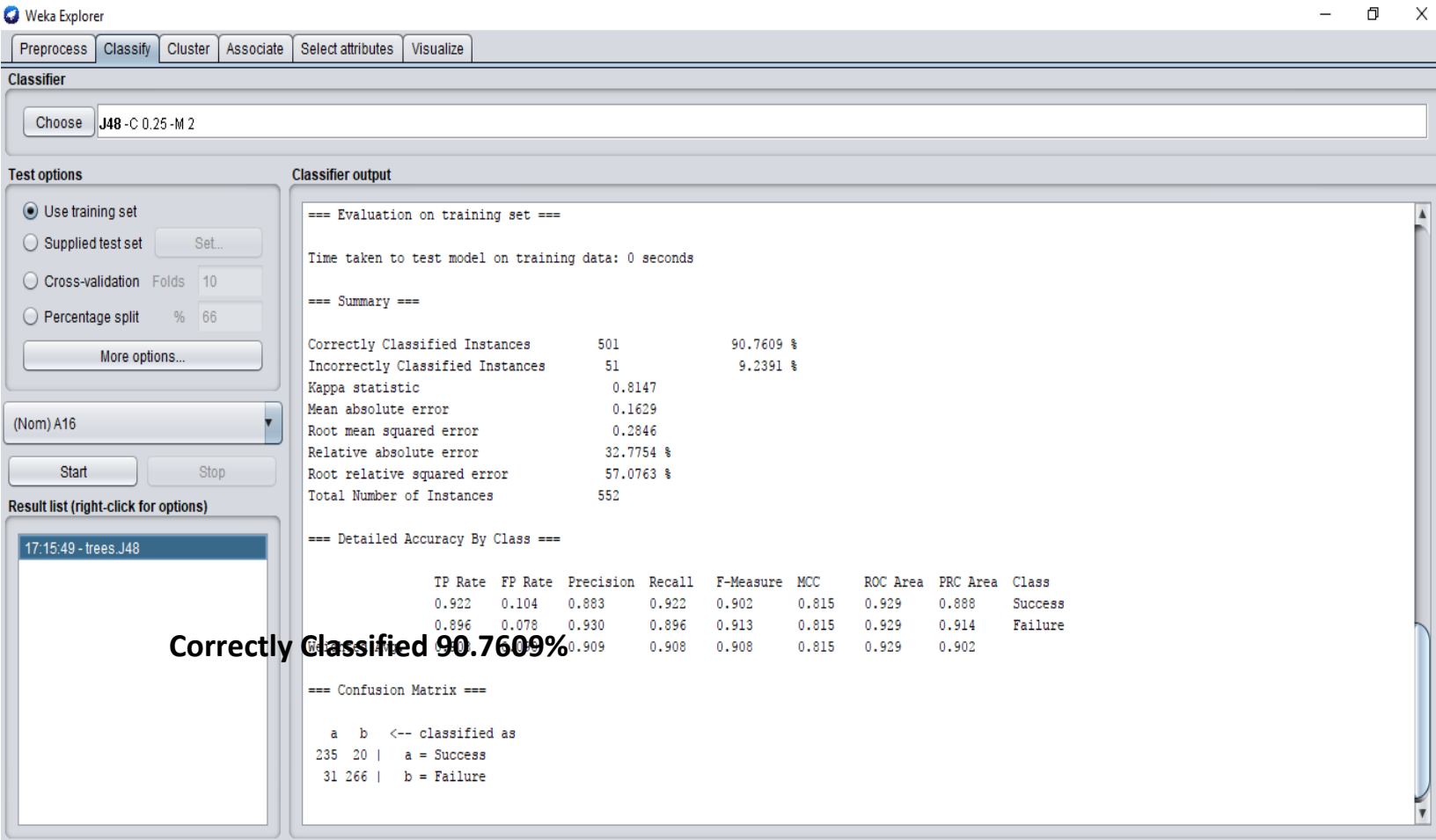


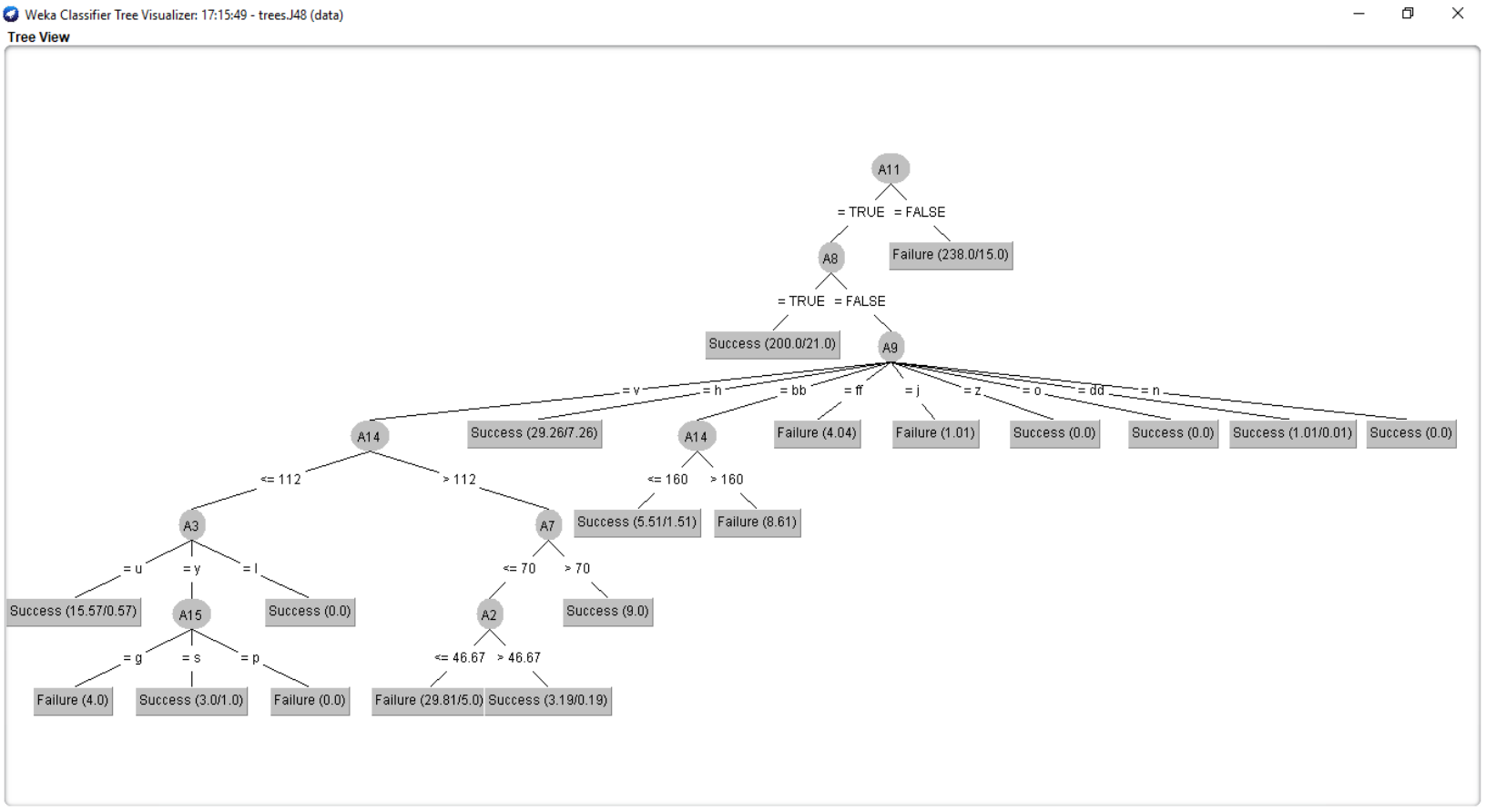
Figure 2: Modified test data in ARFF format

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



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%

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).