Assignment 2

In [1]:
```
## import packages
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from ReliefF import ReliefF
from sklearn.model_selection  import train_test_split
from autorank import autorank
```

In [2]:
```
#1
# read data
data = pd.read_csv('data_A2.csv')
labels = pd.read_csv('labels_A2.csv')
# Fill the missing value with -1
X_fill = np.nan_to_num(data, copy=True, nan=-1)
```

As the missing values scatter among different tuples, so we can't simply delete them. And data itself covers a range of different values so we rule out imputation and instead fill in a global constant to not undermine the variabiliy while remain validilty of the data.

In [3]:
```
#2
#set a random state
RANDOM_STATE = 1234
np.random.seed(RANDOM_STATE)

#pick the 10 most important features using decision tree
dt = DecisionTreeClassifier(random_state=RANDOM_STATE)
dt.fit(X_fill, labels)
fi = dt.feature_importances_
X_clean = X_fill[:,np.argsort(fi)[:10]]
# fs = ReliefF(n_neighbors=10, n_features_to_keep=10)
# X_clean = fs.fit_transform(X_fill, labels)
```

I found them by using the fitted decision tree classifier to calculate the gini importance.

In [4]:
```
#3
df = pd.DataFrame(columns=['RandomForest', 'PrunedDecisionTree', 'UnprunedDec
for x in np.random.rand(100):
    X_train, X_test, y_train, y_test = train_test_split(X_clean, labels.value

    rf = RandomForestClassifier(max_depth=10, random_state= RANDOM_STATE)
    rf.fit(X_train,y_train)

    dt1 = DecisionTreeClassifier(random_state=RANDOM_STATE, ccp_alpha= 0.2)
    dt1.fit(X_train, y_train)

    dt2 = DecisionTreeClassifier(random_state=RANDOM_STATE)
    dt2.fit(X_train, y_train)

    dt3 = DecisionTreeClassifier(random_state=RANDOM_STATE,max_depth=1)
    dt3.fit(X_train, y_train)

    df = df.append({'RandomForest':rf.score(X_test, y_test), 'PrunedDecisionT

result = autorank(df, verbose=False)
print(result)
```

```
RankResult(rankdf=
```

```
                        meanrank    median         mad  ci_lower  ci_upper  \
RandomForest               1.080  0.633503   0.0271872  0.620957  0.646789
UnprunedDecisionTree       2.080  0.577858   0.0277299  0.568584  0.590106
DecisionStump              2.895  0.531812   0.0230604  0.526027  0.551136
PrunedDecisionTree         3.945  0.494960  0.00622718  0.490909  0.498403

                      effect_size   magnitude
RandomForest                    0  negligible
UnprunedDecisionTree      2.02644       large
DecisionStump             4.03403       large
PrunedDecisionTree        7.02479       large
pvalue=1.1964609018114955e-57
cd=0.46903593329832804
omnibus=friedman
posthoc=nemenyi
all_normal=False
pvals_shapiro=[3.85248233314428e-09, 2.6066192225131853e-16, 4.710492618187345
e-08, 4.803963877495236e-15]
homoscedastic=True
pval_homogeneity=0.0648672740107634
homogeneity_test=levene
alpha=0.05
alpha_normality=0.0125
num_samples=100
posterior_matrix=
None
decision_matrix=
None
rope=None
rope_mode=None
effect_size=akinshin_gamma)
```

The prunned decision tree performed the worst as it's overfitting the randomised data and random forest performs the best as it combines features which are not correlated to each other.

In [5]:

```
#4
# additive normal noise
noise = np.random.normal(0, 0.2, np.shape(X_clean))
X_noise = X_clean + np.multiply(noise, np.average(X_train, axis=0))

df = pd.DataFrame(columns=['RandomForest', 'PrunedDecisionTree', 'UnprunedDec
for x in np.random.rand(100):
    X_train, X_test, y_train, y_test = train_test_split(X_noise, labels.value

    rf = RandomForestClassifier(max_depth=10, random_state= RANDOM_STATE)
    rf.fit(X_train,y_train)

    dt1 = DecisionTreeClassifier(random_state=RANDOM_STATE, ccp_alpha= 0.2)
    dt1.fit(X_train, y_train)

    dt2 = DecisionTreeClassifier(random_state=RANDOM_STATE)
    dt2.fit(X_train, y_train)

    dt3 = DecisionTreeClassifier(random_state=RANDOM_STATE,max_depth=1)
    dt3.fit(X_train, y_train)

    df = df.append({'RandomForest':rf.score(X_test, y_test), 'PrunedDecisionT

result = autorank(df, verbose=False)
print(result)
```

```
RankResult(rankdf=
                        meanrank    median         mad  ci_lower  ci_upper  \
RandomForest               1.045  0.627426   0.0312963  0.617347  0.647702
UnprunedDecisionTree       2.135  0.574799   0.0305722  0.561315  0.591241
DecisionStump              2.865  0.541329   0.0198218   0.53405  0.554415
```

```
PrunedDecisionTree       3.955  0.495376  0.00453343  0.492674  0.497449

                     effect_size   magnitude
RandomForest                   0  negligible
UnprunedDecisionTree     1.70113       large
DecisionStump            3.28675       large
PrunedDecisionTree       5.90542       large
pvalue=4.50247885017667e-59
cd=0.46903593329832804
omnibus=friedman
posthoc=nemenyi
all_normal=False
pvals_shapiro=[0.013476056046783924, 1.996535404272701e-16, 9.369413231374857e
-12, 3.637159534264356e-05]
homoscedastic=False
pval_homogeneity=1.1296501742177784e-05
homogeneity_test=levene
alpha=0.05
alpha_normality=0.0125
num_samples=100
posterior_matrix=
None
decision_matrix=
None
rope=None
rope_mode=None
effect_size=akinshin_gamma)
```

The random forest performs signifantly better as the noise can be corrected by combination of features.

In [6]:

```
#5
# multiplicative normal noise
noise = np.random.normal(0, 0.2, np.shape(X_clean))
X_noise = np.multiply(X_clean, noise)

df = pd.DataFrame(columns=['RandomForest', 'PrunedDecisionTree', 'UnprunedDec
for x in np.random.rand(100):
    X_train, X_test, y_train, y_test = train_test_split(X_noise, labels.value

    rf = RandomForestClassifier(max_depth=10, random_state= RANDOM_STATE)
    rf.fit(X_train,y_train)

    dt1 = DecisionTreeClassifier(random_state=RANDOM_STATE, ccp_alpha= 0.2)
    dt1.fit(X_train, y_train)

    dt2 = DecisionTreeClassifier(random_state=RANDOM_STATE)
    dt2.fit(X_train, y_train)

    dt3 = DecisionTreeClassifier(random_state=RANDOM_STATE,max_depth=1)
    dt3.fit(X_train, y_train)

    df = df.append({'RandomForest':rf.score(X_test, y_test), 'PrunedDecisionT

result = autorank(df, verbose=False)
print(result)
```

```
RankResult(rankdf=
                     meanrank    median         mad  ci_lower  ci_upper  \
DecisionStump           2.275  0.497842   0.0163724  0.477833  0.506472
UnprunedDecisionTree    2.440  0.489858   0.0272635  0.477509  0.503145
RandomForest            2.625  0.488859   0.0195547  0.477912       0.5
PrunedDecisionTree      2.660  0.494109  0.00583907  0.483553  0.496454

                     effect_size   magnitude
DecisionStump                  0  negligible
UnprunedDecisionTree    0.355036       small
```

```
RandomForest            0.498115       small
PrunedDecisionTree      0.303658       small
pvalue=0.11760300259309563
cd=0.46903593329832804
omnibus=friedman
posthoc=nemenyi
all_normal=False
pvals_shapiro=[5.187140232010279e-06, 3.3028347771353053e-16, 2.81377447208797
1e-06, 3.8545308302584055e-14]
homoscedastic=True
pval_homogeneity=0.33494209462884944
homogeneity_test=levene
alpha=0.05
alpha_normality=0.0125
num_samples=100
posterior_matrix=
None
decision_matrix=
None
rope=None
rope_mode=None
effect_size=akinshin_gamma)
```

All the classifiers are worsen in this case as the noise can't be corrected due to randomised scalar.

In [7]:

```
#7
#Adding noise to trainning set
df = pd.DataFrame(columns=['RandomForest', 'PrunedDecisionTree', 'UnprunedDec
for x in np.random.rand(100):
    X_train, X_test, y_train, y_test = train_test_split(X_clean, labels.value

    noise = np.random.normal(0, 0.2, np.shape(X_train))
    X_train = np.multiply(X_train, noise)

    rf = RandomForestClassifier(max_depth=10, random_state= RANDOM_STATE)
    rf.fit(X_train,y_train)

    dt1 = DecisionTreeClassifier(random_state=RANDOM_STATE, ccp_alpha= 0.2)
    dt1.fit(X_train, y_train)

    dt2 = DecisionTreeClassifier(random_state=RANDOM_STATE)
    dt2.fit(X_train, y_train)

    dt3 = DecisionTreeClassifier(random_state=RANDOM_STATE,max_depth=1)
    dt3.fit(X_train, y_train)

    df = df.append({'RandomForest':rf.score(X_test, y_test), 'PrunedDecisionT

result = autorank(df, verbose=False)
print(result)

#Adding noise to test set
df = pd.DataFrame(columns=['RandomForest', 'PrunedDecisionTree', 'UnprunedDec
for x in np.random.rand(100):
    X_train, X_test, y_train, y_test = train_test_split(X_clean, labels.value

    noise = np.random.normal(0, 0.2, np.shape(X_test))
    X_test = np.multiply(X_test, noise)

    rf = RandomForestClassifier(max_depth=10, random_state= RANDOM_STATE)
    rf.fit(X_train,y_train)

    dt1 = DecisionTreeClassifier(random_state=RANDOM_STATE, ccp_alpha= 0.2)
    dt1.fit(X_train, y_train)
```

```
    dt2 = DecisionTreeClassifier(random_state=RANDOM_STATE)
    dt2.fit(X_train, y_train)

    dt3 = DecisionTreeClassifier(random_state=RANDOM_STATE,max_depth=1)
    dt3.fit(X_train, y_train)

    df = df.append({'RandomForest':rf.score(X_test, y_test), 'PrunedDecisionT

result = autorank(df, verbose=False)
print(result)
```

```
/Users/channingwang/opt/anaconda3/lib/python3.8/site-packages/scipy/stats/more
stats.py:1678: UserWarning: Input data for shapiro has range zero. The results
may not be accurate.
  warnings.warn("Input data for shapiro has range zero. The results "
RankResult(rankdf=
                      meanrank  median       mad ci_lower ci_upper  \
RandomForest             1.535   0.550  0.044478     0.53     0.57
UnprunedDecisionTree     2.075   0.515  0.051891     0.49     0.54
DecisionStump            2.460   0.480  0.088956     0.43     0.55
PrunedDecisionTree       3.930   0.400         0      0.4      0.4

                     effect_size   magnitude
RandomForest                   0  negligible
UnprunedDecisionTree    0.724235      medium
DecisionStump           0.995366       large
PrunedDecisionTree       4.76937       large
pvalue=1.260484602570955e-41
cd=0.46903593329832804
omnibus=friedman
posthoc=nemenyi
all_normal=False
pvals_shapiro=[0.05064091831445694, 1.0, 0.33107009530067444, 2.10458182436923
38e-07]
homoscedastic=False
pval_homogeneity=2.789510644309798e-47
homogeneity_test=levene
alpha=0.05
alpha_normality=0.0125
num_samples=100
posterior_matrix=
None
decision_matrix=
None
rope=None
rope_mode=None
effect_size=akinshin_gamma)
RankResult(rankdf=
                      meanrank  median       mad ci_lower ci_upper  \
RandomForest             1.440    0.48  0.059304     0.46     0.51
UnprunedDecisionTree     1.815    0.45  0.044478     0.43     0.48
DecisionStump            3.230    0.40         0      0.4     0.41
PrunedDecisionTree       3.515    0.40         0      0.4      0.4

                     effect_size   magnitude
RandomForest                   0  negligible
UnprunedDecisionTree    0.572324      medium
DecisionStump            1.90775       large
PrunedDecisionTree       1.90775       large
pvalue=3.2470568055732433e-44
cd=0.46903593329832804
omnibus=friedman
posthoc=nemenyi
all_normal=False
pvals_shapiro=[0.035578273236751556, 1.0, 0.031193194910883904, 2.212469617235
424e-11]
homoscedastic=False
pval_homogeneity=4.6189838773291195e-50
```

```
homogeneity_test=levene
alpha=0.05
alpha_normality=0.0125
num_samples=100
posterior_matrix=
None
decision_matrix=
None
rope=None
rope_mode=None
effect_size=akinshin_gamma)
```

```
/Users/channingwang/opt/anaconda3/lib/python3.8/site-packages/scipy/stats/morestats.py:1678: UserWarning: Input data for shapiro has range zero. The results may not be accurate.
  warnings.warn("Input data for shapiro has range zero. The results "
```

Adding the noise into the test set has a bigger impact as we base on a wrong set to train the model.