In [1250]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import Perceptron
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import ExtraTreesClassifier
```

In [1251]:
```
# Read in the dataset using Pandas into a dataframe
df = pd.read_csv("http://archive.ics.uci.edu/ml/machine-learning-databas
                 names = ["Class Name", "Left-Weight", "Left-Distance",
```

In [1252]:
```
df.info()
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 625 entries, 0 to 624
Data columns (total 5 columns):
Class Name        625 non-null object
Left-Weight       625 non-null int64
Left-Distance     625 non-null int64
Right-Weight      625 non-null int64
Right-Distance    625 non-null int64
dtypes: int64(4), object(1)
memory usage: 24.5+ KB
```

Out[1252]:

| | Class Name | Left-Weight | Left-Distance | Right-Weight | Right-Distance |
|---|---|---|---|---|---|
| **0** | B | 1 | 1 | 1 | 1 |
| **1** | R | 1 | 1 | 1 | 2 |
| **2** | R | 1 | 1 | 1 | 3 |
| **3** | R | 1 | 1 | 1 | 4 |
| **4** | R | 1 | 1 | 1 | 5 |

In [1253]:
```
# The data has categorical lablels in the Class Name column. Converting
df = df.replace('L',1)
df = df.replace('B',2)
df = df.replace('R',3)
```

In [1254]:
```
balance = df.values
```

In [1255]:
```
X,y = balance[:,1:5], balance[:,0:1]
X,y = X.astype(float), y.astype(float)
```

In [1256]:
```
scaler = StandardScaler()
```

In [1257]:
```
scaler.fit(X)
```

Out[1257]:
```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

In [1258]:
```
X = scaler.transform(X)
modelForAttributeImportance = ExtraTreesClassifier()
modelForAttributeImportance.fit(X, y.ravel())
print(modelForAttributeImportance.feature_importances_)
```

```
[ 0.25107147  0.25240886  0.25411232  0.24240735]
```

In [1259]:
```
models = []
```

In [1260]:
```
models.append(('Decision Tree', DecisionTreeClassifier(
 criterion="entropy",
 splitter="best",
 max_depth=40,
 min_samples_split=3,
 min_samples_leaf=5,
 min_weight_fraction_leaf=0.,
 max_features=None,
 random_state=None,
 max_leaf_nodes=None,
 min_impurity_decrease=0.,
 min_impurity_split=None,
 class_weight=None,
 presort=False)))
```

In [1261]:
```
models.append(('Perceptron', Perceptron(
penalty='l1',
alpha=0.0002,
fit_intercept=True,
max_iter=1500,
tol=0.0,
shuffle=False,
verbose=0,
eta0=1.0,
n_jobs=1,
random_state=None,
class_weight=None,
warm_start=False,
n_iter=None)))
```

In [1262]:
```
models.append(('Neural Net', MLPClassifier(
hidden_layer_sizes=(8,3),
activation="logistic",
solver='lbfgs',
alpha=1e-5,
batch_size='auto',
learning_rate="constant",
learning_rate_init=0.004,
power_t=0.5,
max_iter=1000,
shuffle=True,
random_state=None,
tol=1e-4,
verbose=False,
warm_start=False,
momentum=0.9,
nesterovs_momentum=True,
early_stopping=False,
validation_fraction=0.1,
beta_1=0.9, beta_2=0.999,
epsilon=1e-8)))
```

In [1263]:
```
models.append(('Deep Learning', MLPClassifier(
hidden_layer_sizes=(10,10,10),
activation="relu",
solver='lbfgs',
alpha=1e-5,
batch_size='auto',
learning_rate="constant",
learning_rate_init=0.004,
power_t=0.5,
max_iter=1000,
shuffle=True,
random_state=None,
tol=1e-4,
verbose=False,
warm_start=False,
momentum=0.9,
nesterovs_momentum=True,
early_stopping=False,
validation_fraction=0.1,
beta_1=0.9, beta_2=0.999,
epsilon=1e-8)))
```

In [1264]:
```
models.append(('SVM', SVC(
C=1.0,
kernel='sigmoid',
degree=2,
gamma='auto',
coef0=0.0,
shrinking=True,
probability=False,
tol=1e-3,
cache_size=200,
class_weight=None,
verbose=False,
max_iter=-1,
decision_function_shape='ovo',
random_state=None)))
```

In [1265]:
```
models.append(('Naive Bayes', GaussianNB(priors=None)))
```

In [1266]:
```
models.append(('Logistic Regression', LogisticRegression(
penalty='l2',
dual=False,
tol=1e-4,
C=1.0,
fit_intercept=True,
intercept_scaling=1,
class_weight=None,
random_state=None,
solver='newton-cg',
max_iter=200,
multi_class='multinomial',
verbose=0,
warm_start=False,
n_jobs=1)))
```

In [1267]:
```
models.append(('K Neighbors Classifier', KNeighborsClassifier(
n_neighbors=5,
weights='distance',
algorithm='auto',
leaf_size=30,
p=1,
metric='minkowski',
metric_params=None,
n_jobs=1)))
```

In [1268]:
```
models.append(('Bagging Classifier', BaggingClassifier(
base_estimator=None,
n_estimators=14,
max_samples=1.0,
max_features=1.0,
bootstrap=True,
bootstrap_features=False,
oob_score=False,
warm_start=False,
n_jobs=1,
random_state=None,
verbose=0)))
```

In [1269]:
```
models.append(('Random Forest Classifier', RandomForestClassifier(
n_estimators=10,
criterion="entropy",
max_depth=10,
min_samples_split=4,
min_samples_leaf=1,
min_weight_fraction_leaf=0.,
max_features="auto",
max_leaf_nodes=None,
min_impurity_decrease=0.,
min_impurity_split=None,
bootstrap=True,
oob_score=False,
n_jobs=1,
random_state=None,
verbose=0,
warm_start=False,
class_weight=None)))
```

In [1270]:
```
models.append(('AdaBoost Classifier', AdaBoostClassifier(
base_estimator=None,
n_estimators=50,
learning_rate=2.,
algorithm='SAMME',
random_state=None)))
```

In [1271]:
```
models.append(('Gradient Boosting Classifier', GradientBoostingClassifie
loss='deviance',
learning_rate=0.2,
n_estimators=100,
subsample=1.0,
criterion='friedman_mse',
min_samples_split=2,
min_samples_leaf=1,
min_weight_fraction_leaf=0.,
max_depth=2,
min_impurity_decrease=0.,
min_impurity_split=None,
init=None,
random_state=None,
max_features=None,
verbose=0,
max_leaf_nodes=None,
warm_start=False,
presort='auto')))
```

In [1272]:
```
results = []
names = []
scoring = 'accuracy'
print("Average accuracy of all models:\n")
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=0)
    cv_results = model_selection.cross_val_score(model, X, y.ravel(), cv
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f %f" % (name, cv_results.mean(), cv_results.std() )
    print(msg)
```

```
Average accuracy of all models:

Decision Tree: 0.774731 0.060409
Perceptron: 0.857783 0.053205
Neural Net: 0.928111 0.046207
Deep Learning: 0.964823 0.028507
SVM: 0.881644 0.029594
Naive Bayes: 0.878469 0.025634
Logistic Regression: 0.856068 0.033967
K Neighbors Classifier: 0.854736 0.070027
Bagging Classifier: 0.806528 0.032354
Random Forest Classifier: 0.813031 0.046921
AdaBoost Classifier: 0.852944 0.076056
Gradient Boosting Classifier: 0.891347 0.035077
```

In [1273]:
```
results2 = []
names = []
print("Average precision of all models:\n")
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=0)
    cv_results2 = model_selection.cross_val_score(model, X, y.ravel(), c
    results2.append(cv_results2)
    names.append(name)
    msg2 = "%s: %f" % (name, cv_results2.mean())
    print(msg2)
```

```
Average precision of all models:

Decision Tree: 0.773118
Perceptron: 0.857783
Neural Net: 0.950512
Deep Learning: 0.969688
SVM: 0.881644
Naive Bayes: 0.878469
Logistic Regression: 0.856068
K Neighbors Classifier: 0.854736
Bagging Classifier: 0.809831
Random Forest Classifier: 0.822837
AdaBoost Classifier: 0.852944
Gradient Boosting Classifier: 0.892960
```