Here is a list of 10 Essential Algorithms that you should know to understand the basics of [**#MachineLearning**](https://www.linkedin.com/uas/login?session_redirect=https%3A%2F%2Fwww.linkedin.com%2Ffeed%2Fhashtag%2Fmachinelearning&trk=public-post_share-update_update-text): - - -

|  |  |
| --- | --- |
| Logistic Regression | https://lnkd.in/gJ2BwhD |
| Linear Regression | https://lnkd.in/gdZDbT5 |
| Decision Trees | https://lnkd.in/gwadA-p |
| Random Forests | https://lnkd.in/gRYHcvt |
| Neural Networks | https://lnkd.in/gZQhWyv |
| Bayesian Techniques | https://lnkd.in/gY3qVYP |
| Support Vector Machines | https://lnkd.in/gWJKRyn |
| Gradient Boosting Machine | https://lnkd.in/gv85yDV |
| K-Nearest Neighbors | https://lnkd.in/gsiyqcM |
| Regularized Linear Models | https://lnkd.in/g3fn3c |

**import** **numpy** **as** **np**

**import** **pandas** **as** **pd**

**from** **sklearn.model\_selection** **import** train\_test\_split

**from** **sklearn** **import** preprocessing

*# load tic-tac-toe data*

data = pd.read\_csv("tic-tac-toe.data", sep=",")

data.rename(columns={'x': 'top left', 'x.1': 'top middle', 'x.2': 'top right','x.3': 'middle left', 'o': 'middle middle', 'o.1' : 'middle right', 'x.4' : 'bottom left', 'o.2' : 'bottom middle', 'o.3':'bottom right','positive' : 'outcome'},inplace=**True**)

data.head(3)

data\_final = pd.concat([data\_new, data.ix[:,9]],axis=1)

*# Split data into training and test sets*

train, test = train\_test\_split(data\_final, test\_size=0.3)

train.head()

*#Applying logistic Regression model*

**from** **sklearn.linear\_model** **import** LogisticRegression

LR = LogisticRegression()

x\_train = train.ix[:, :-1]

y\_train= train.ix[:, -1]

x\_test = test.ix[:, :-1]

y\_test= test.ix[:, -1]

LR = LR.fit(x\_train, y\_train)

LR.score(x\_train, y\_train)

*#Evaluate model on test data*

**from** **sklearn** **import** metrics

**from** **sklearn.cross\_validation** **import** cross\_val\_score

prob = LR.predict\_proba(x\_test)

predicted= LR.predict(x\_test)

*#Findind the accuracy using confusion matrix*

print(metrics.accuracy\_score(y\_test, predicted))

print(metrics.confusion\_matrix(y\_test, predicted))

print(metrics.classification\_report(y\_test, predicted))

*#Finding the accuracy using cross validation method*

scores = cross\_val\_score(LogisticRegression(), data\_final.ix[:, 0:27], data\_final.ix[:, 27], scoring='accuracy', cv=10)

print(scores)

print(scores.mean())

**Random Forest Classifier**

Applies **weight concept** → to consider the impact of result from any decision tree.

Lower weight – DT with high error rate

Higher weight – DT with low error rate