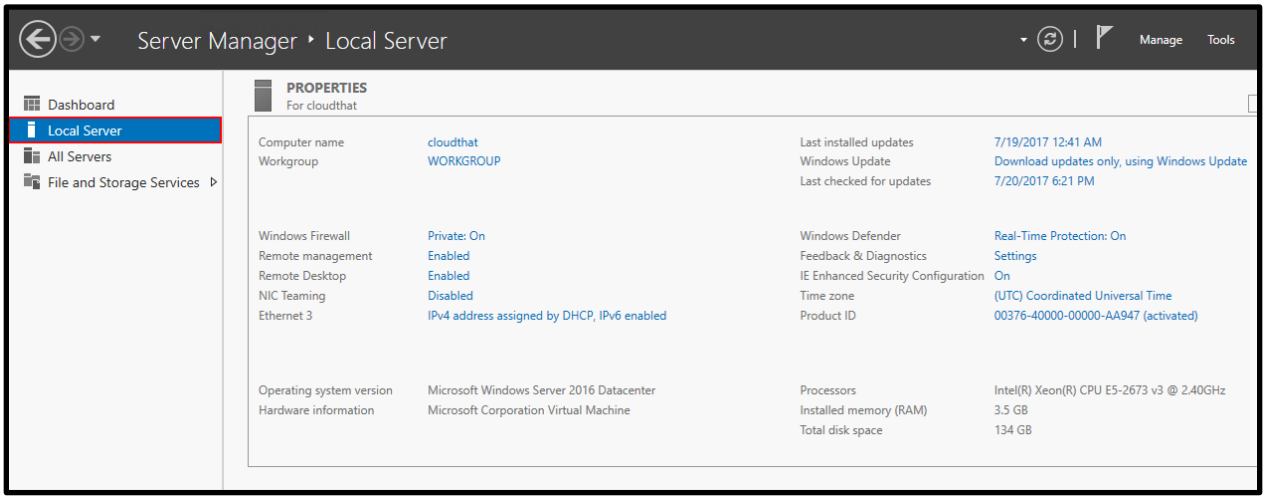
# Lab: Logistic Regression with Python

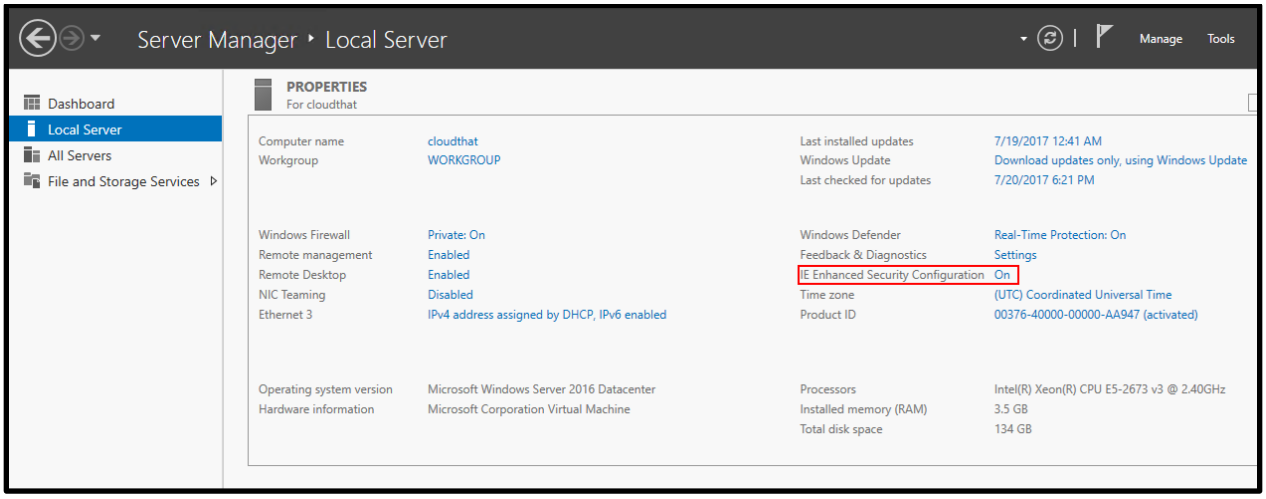
- Set up the environment
- Install Anaconda
- Performing the experiment to compare various Models

## Task 1: Set up the environment

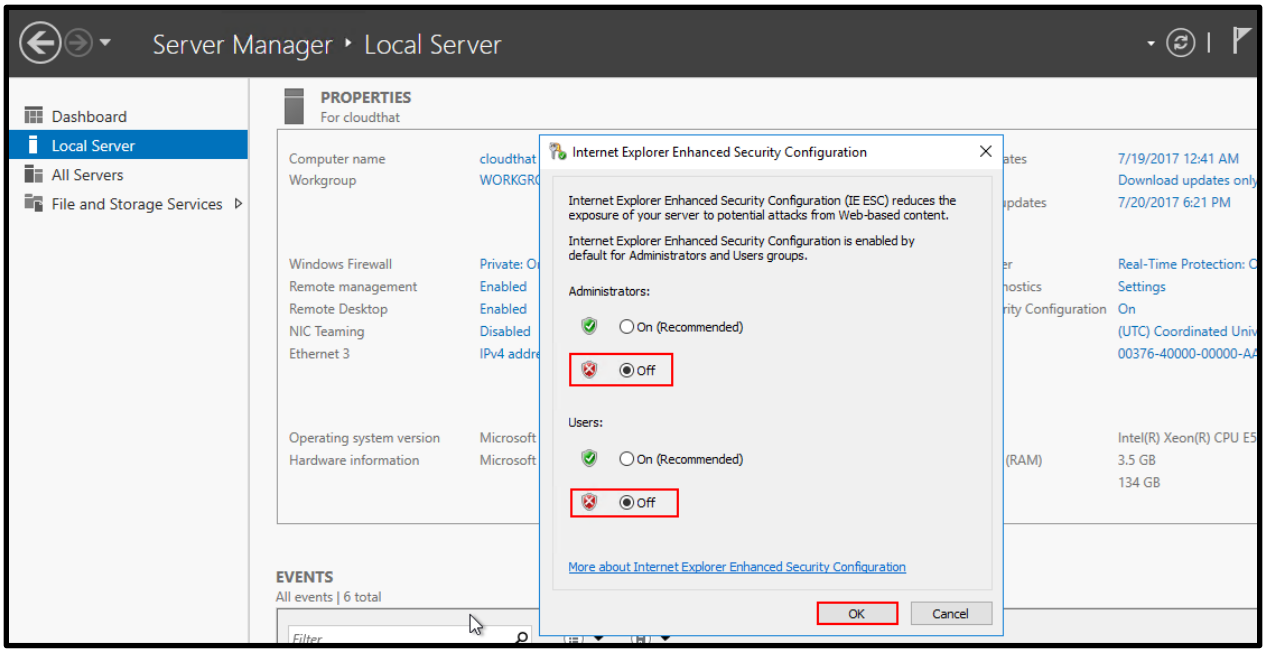
1. Create a VM "Windows Server 2016 Datacenter" in Azure
2. Click on **Local Server**




3. Click on **on** to modify the configuration

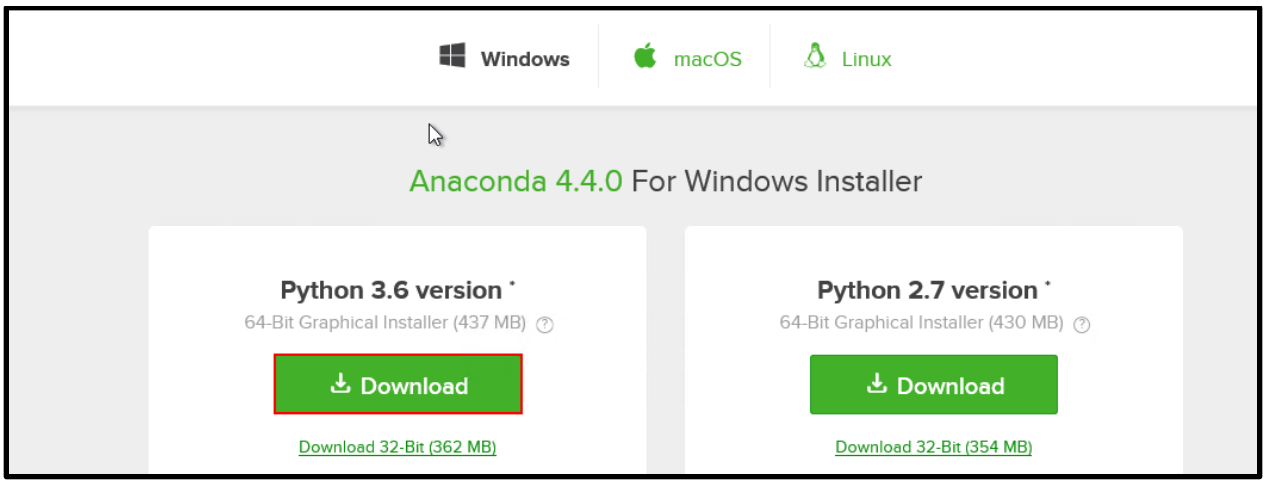


4. Click on **off** as shown in image below and click **ok** later

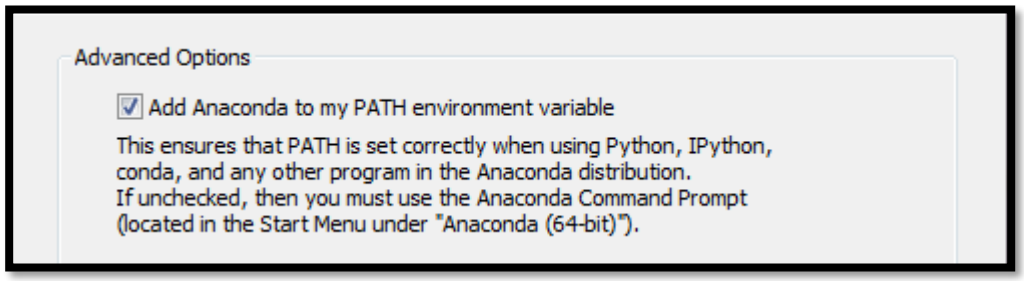



## Task2: Install Anacondas 3

1. Open a browser and go to the following address https://www.anaconda.com/download/ and download the anacondas for python3.6

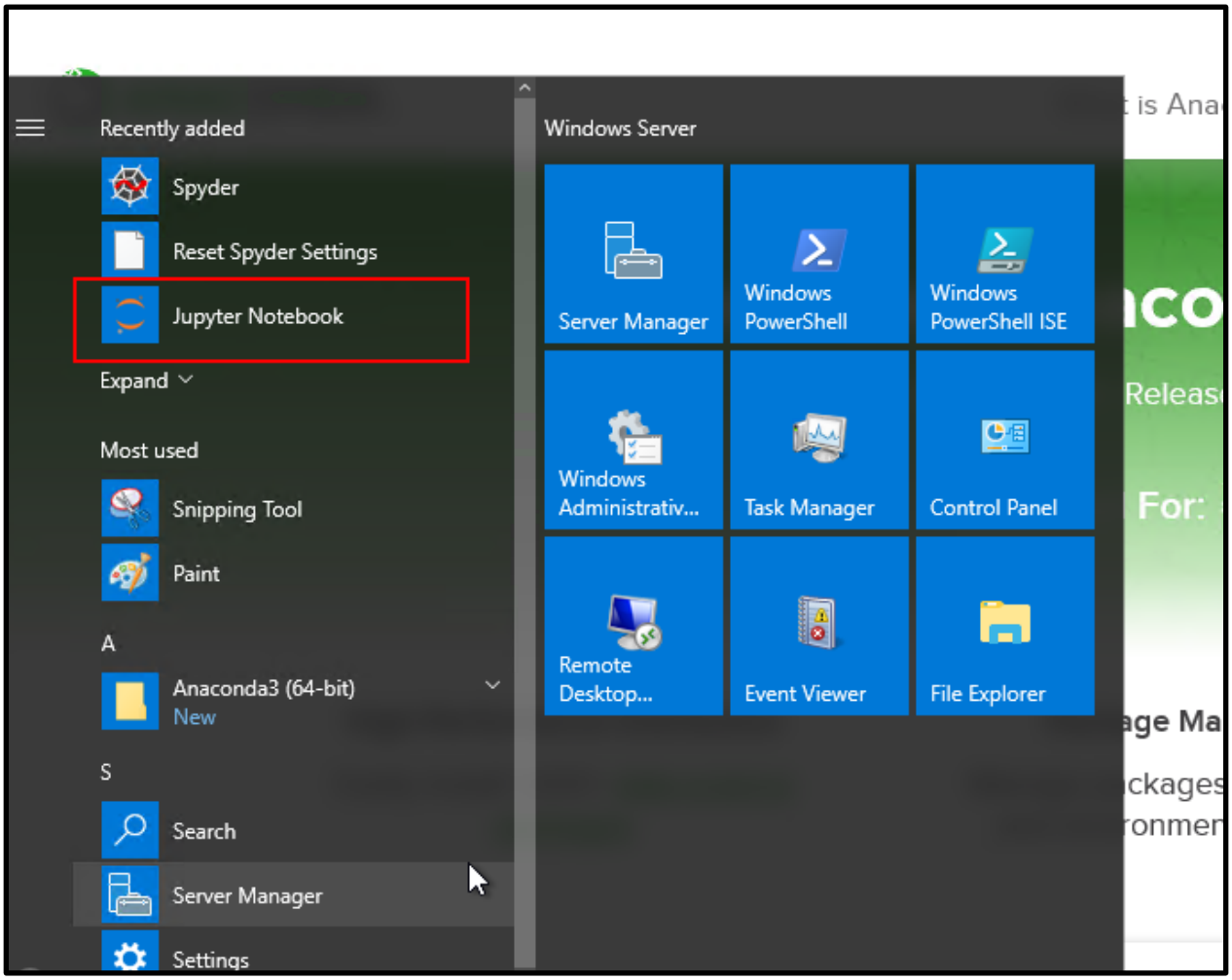


2. Run as per instructions, click **Next**




3. Note: Select Add Anaconda to my PATH environment variable as well, let other settings default click on Install

4. Press windows key and select **Jupyter Notebook**



## Task 3: Performing the experiment to compare various Models

1. Click on **New** and select **Python3** and add the following code to the cell for checking the environment

```
# Python version
import sys
print('Python: {}'.format(sys.version))
# scipy
import scipy
print('scipy: {}'.format(scipy.__version__))
# numpy
import numpy
print('numpy: {}'.format(numpy.__version__))
# matplotlib
import matplotlib
print('matplotlib: {}'.format(matplotlib.__version__))
# pandas
import pandas
print('pandas: {}'.format(pandas.__version__))
# scikit-learn
import sklearn
print('sklearn: {}'.format(sklearn.__version__))
```

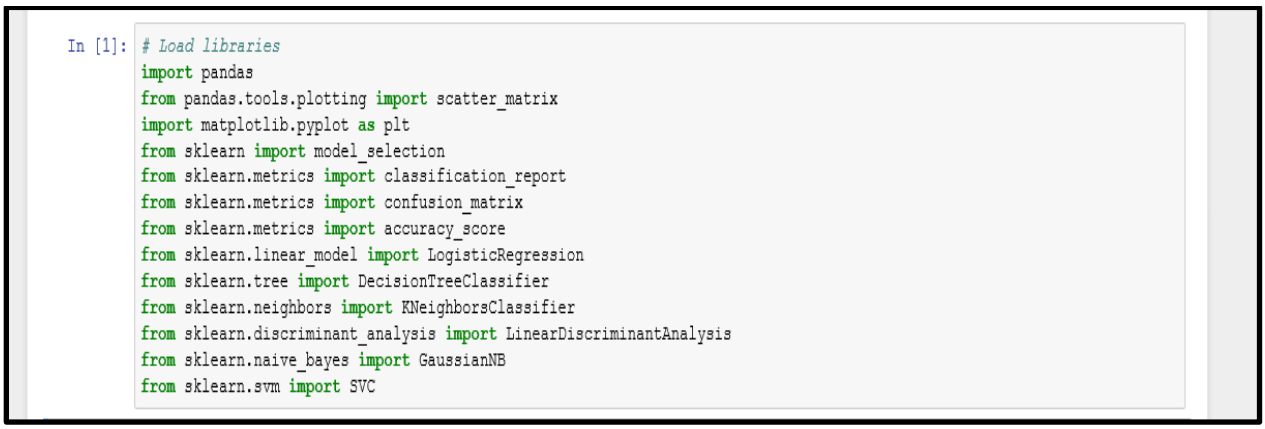
```
In [ ]: # Check the versions of libraries

        # Python version
        import sys
        print('Python: {}'.format(sys.version))
        # scipy
        import scipy
        print('scipy: {}'.format(scipy.__version__))
        # numpy
        import numpy
        print('numpy: {}'.format(numpy.__version__))
        # matplotlib
        import matplotlib
        print('matplotlib: {}'.format(matplotlib.__version__))
        # pandas
        import pandas
        print('pandas: {}'.format(pandas.__version__))
        # scikit-learn
        import sklearn
        print('sklearn: {}'.format(sklearn.__version__))
```

2. If you see following output, then environment is configured in right way

```
Python: 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11 2017, 13:25:24) [MSC v.1900 64 bit (AMD64)]
scipy: 0.19.0
numpy: 1.12.1
matplotlib: 2.0.2
pandas: 0.20.1
sklearn: 0.18.1
```

3. Importing all the required libraries

```
# Load libraries
import pandas
from pandas.tools.plotting import scatter_matrix
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
```




4. Load the Dataset

```
# Load dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
dataset = pandas.read_csv(url, names=names)
```

```
In [2]: # Load dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
dataset = pandas.read_csv(url, names=names)
```

5. Dimension of the dataset, run the following command and observe the dataset.

```
# shape
print(dataset.shape)
```

```
In [4]: # shape
print(dataset.shape)

(150, 5)
```

6. See all the top 20 rows of dataset

```
# head
print(dataset.head(20))
```

```
In [5]: # head
print(dataset.head(20))

    sepal-length  sepal-width  petal-length  petal-width        class
0            5.1          3.5           1.4          0.2  Iris-setosa
1            4.9          3.0           1.4          0.2  Iris-setosa
2            4.7          3.2           1.3          0.2  Iris-setosa
3            4.6          3.1           1.5          0.2  Iris-setosa
4            5.0          3.6           1.4          0.2  Iris-setosa
5            5.4          3.9           1.7          0.4  Iris-setosa
6            4.6          3.4           1.4          0.3  Iris-setosa
7            5.0          3.4           1.5          0.2  Iris-setosa
8            4.4          2.9           1.4          0.2  Iris-setosa
9            4.9          3.1           1.5          0.1  Iris-setosa
10           5.4          3.7           1.5          0.2  Iris-setosa
11           4.8          3.4           1.6          0.2  Iris-setosa
12           4.8          3.0           1.4          0.1  Iris-setosa
13           4.3          3.0           1.1          0.1  Iris-setosa
14           5.8          4.0           1.2          0.2  Iris-setosa
15           5.7          4.4           1.5          0.4  Iris-setosa
16           5.4          3.9           1.3          0.4  Iris-setosa
17           5.1          3.5           1.4          0.3  Iris-setosa
18           5.7          3.8           1.7          0.3  Iris-setosa
19           5.1          3.8           1.5          0.3  Iris-setosa
```

7. View some of the statistical properties of the dataset like mean, count, etc.

```
# descriptions
print(dataset.describe())
```

```
In [6]: # descriptions
        print(dataset.describe())

               sepal-length  sepal-width  petal-length  petal-width
count        150.000000   150.000000    150.000000   150.000000
mean           5.843333     3.054000      3.758667     1.198667
std            0.828066     0.433594      1.764420     0.763161
min            4.300000     2.000000      1.000000     0.100000
25%            5.100000     2.800000      1.600000     0.300000
50%            5.800000     3.000000      4.350000     1.300000
75%            6.400000     3.300000      5.100000     1.800000
max            7.900000     4.400000      6.900000     2.500000
```

8. Number of instances (rows) belonging to each class

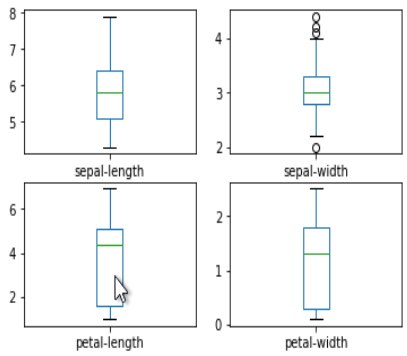
```
# class distribution
print(dataset.groupby('class').size())
```

```
In [7]: # class distribution
        print(dataset.groupby('class').size())

class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

9. Visualizing the data, building an univariate plot

```
# box and whisker plots
dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False,
sharey=False)
plt.show()
```

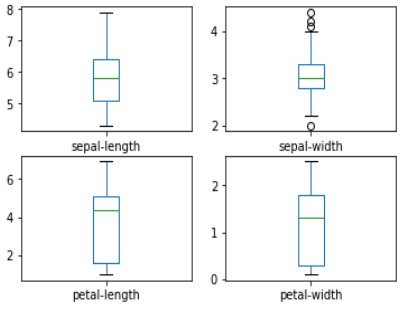
```
In [8]: # box and whisker plots
        dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
        plt.show()
```




10. Visualizing the data, plotting a histogram

```
# histograms
dataset.hist()
plt.show()
```

```
In [8]: # box and whisker plots
        dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
        plt.show()
```

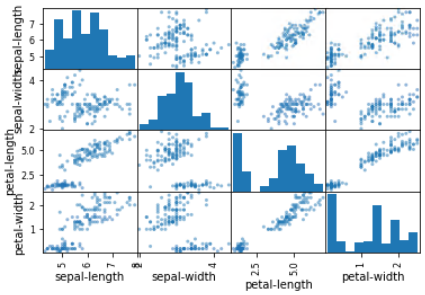


11. Visualizing the data, multivariate plot

```
# scatter plot matrix
scatter_matrix(dataset)
plt.show()
```

```
In [9]: # scatter plot matrix
        scatter_matrix(dataset)
        plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: FutureWarning: 'pandas.tools.plotting.scatter_matrix' is deprecated, import 'pandas.plotting.scatter_matrix' instead.
```




12. Split the data-set

```
# Split-out validation dataset
array = dataset.values
X = array[:,0:4]
Y = array[:,4]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation =
model_selection.train_test_split(X, Y, test_size=validation_size,
random_state=seed)
```

```
In [10]: # Split-out validation dataset
         array = dataset.values
         X = array[:,0:4]
         Y = array[:,4]
         validation_size = 0.20
         seed = 7
         X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, Y, test_size=validation_size, random_
```

13. Setting the parameters for estimating accuracy

# test options and evaluation metric
Seed = 7
Scoring = ‘accuracy’

```
In [11]: # Test options and evaluation metric
         seed = 7
         scoring = 'accuracy'
```

14. Build all the required models

```
# Spot Check Algorithms
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
# evaluate each model in turn
results = []
names = []
for name, model in models:
        kfold = model_selection.KFold(n_splits=10, random_state=seed)
        cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold,
scoring=Scoring)
        results.append(cv_results)
        names.append(name)
        msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
        print(msg)
```

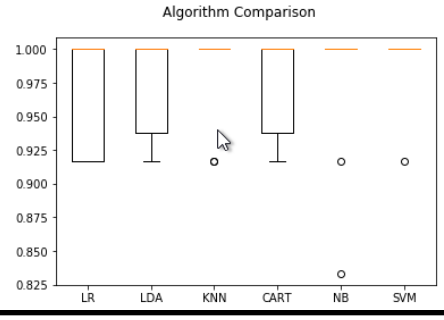
In [12]:
```
# Spot Check Algorithms
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed)
    cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
LR: 0.966667 (0.040825)
LDA: 0.975000 (0.038188)
KNN: 0.983333 (0.033333)
CART: 0.975000 (0.038188)
NB: 0.975000 (0.053359)
SVM: 0.991667 (0.025000)
```

15. Select the best model out of it

```
# Compare Algorithms
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

In [13]:
```
# Compare Algorithms
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```



Making Prediction on validation dataset

```
# Make predictions on validation dataset
lr = LogisticRegression()
lr.fit(X_train, Y_train)
predictions = lr.predict(X_validation)
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
```

In [18]:

```
# Make predictions on validation dataset
lr = LogisticRegression()
lr.fit(X_train, Y_train)
predictions = lr.predict(X_validation)
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
```

```
0.8
[[ 7  0  0]
 [ 0  7  5]
 [ 0  1 10]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       0.88      0.58      0.70        12
 Iris-virginica       0.67      0.91      0.77        11

    avg / total       0.83      0.80      0.80        30
```