

Configuration Manual

MSc Research Project

Data Analytics

Sreenand Kandath

Student ID: x18137636

School of Computing

National College of Ireland

Supervisor: Dr. Catherine Mulwa

National College of Ireland
MSc Project Submission Sheet
School of Computing

Student Name : Sreenand Kandath

Student ID : X18137636

Programme : MSc Data Analytics

Year: 2019-2020

Module : MSc Data Analytics - Research Project

Supervisor : Dr. Catherine Mulwa

Submission Due 13/12/2019
Date:

Word Count : 1447 words

Page Count : 23

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature
:

Date:
.....

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission , to each project (including multiple copies).	<input type="checkbox"/>
You must ensure that you retain a HARD COPY of the project , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Sreenand Kandath
Student ID: X18137636

1 Introduction

The configurations manual illustrates the environment setup and configuration parameters which was made available for the Msc Research project “Air Quality Quantification in Taiwan Using Machine Learning Techniques in Apache Spark” .

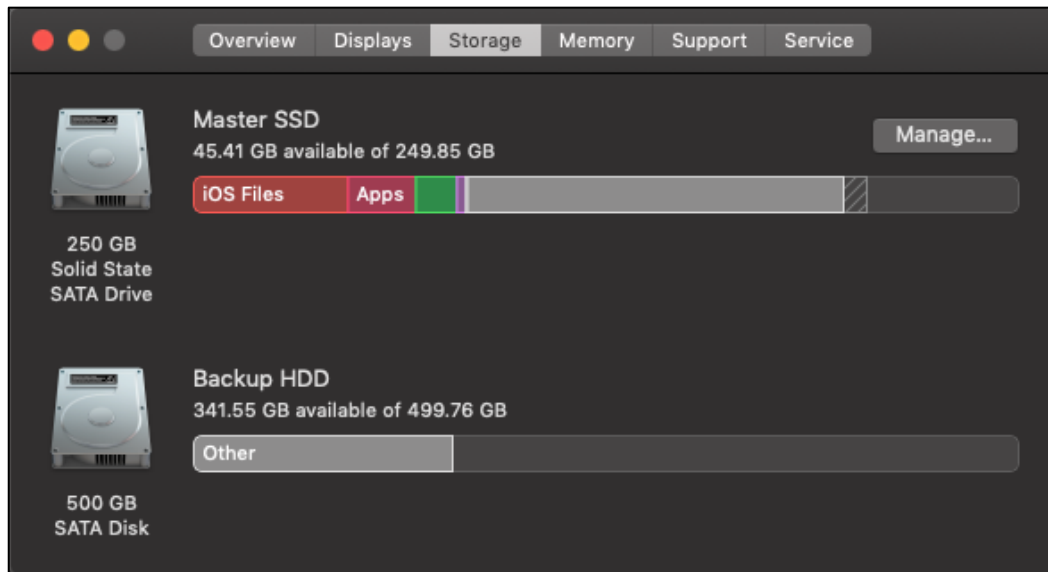
2 Environment Specification and Configuration

2.1 Hardware Configuration

The hardware configuration used for this project is mentioned below with a snapshot of the system details in figure 1 and figure 2.

- MacBook Pro (13-inch, Mid 2012)
- Processor : 2.5 GHz Intel Core i5
- Memory : 16 GB 1600 MHz DDR3
- Storage : 250gb SSD, 500gb HDD
- Graphics : Intel HD Graphics 4000 1536 MB





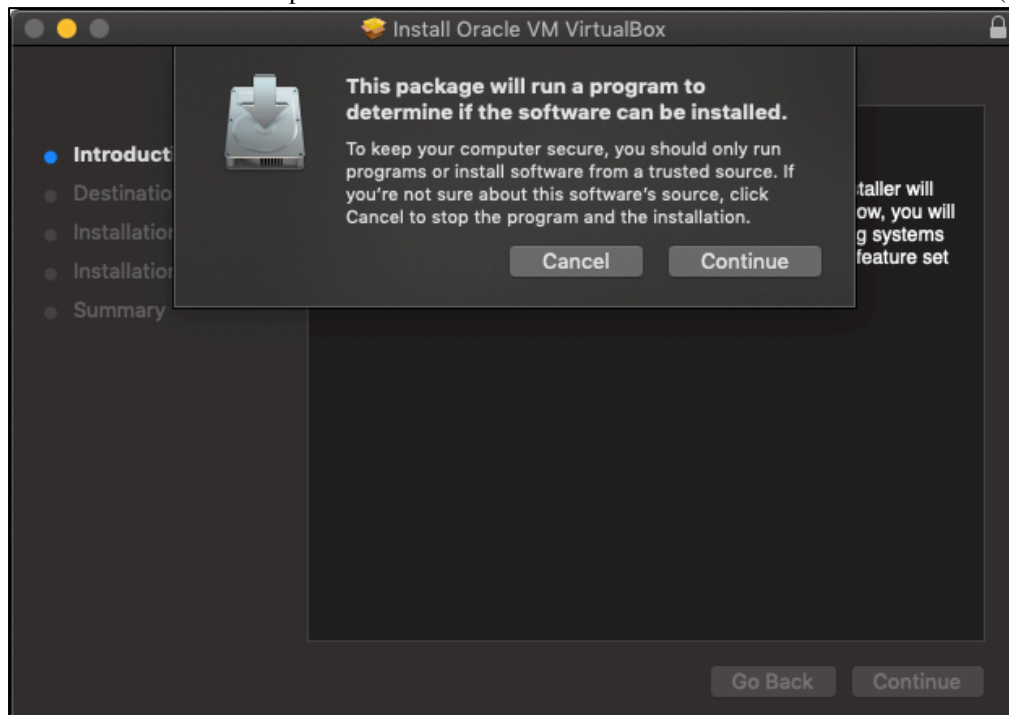
2.2 Software Specifications and Requirements

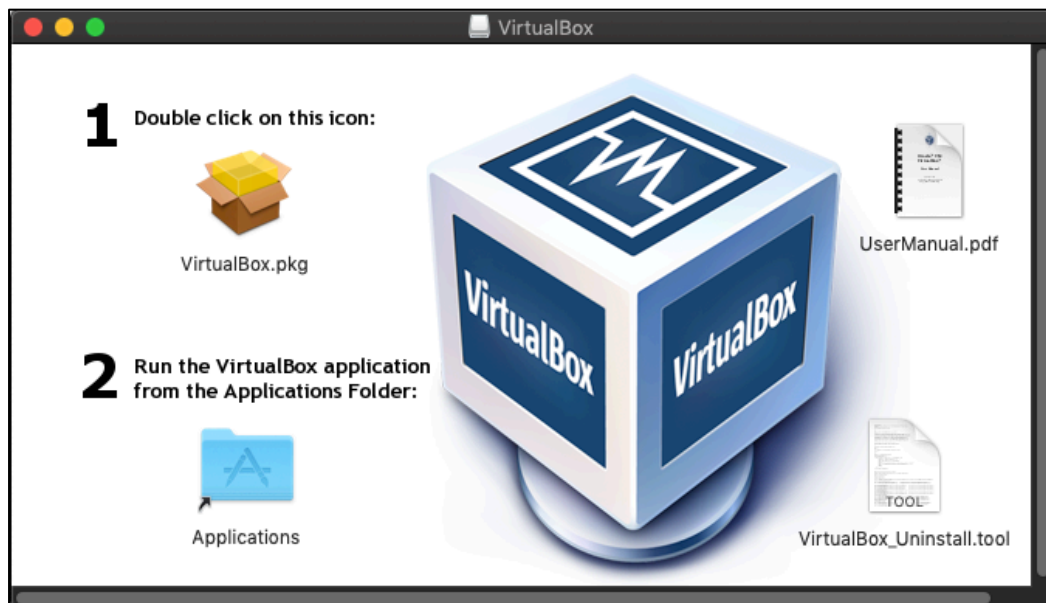
The basic operating system used for this research is the macOS Mojave. To implement the Apache Spark ecosystem, VirtualBox application was installed with ubuntu operating system. The ubuntu operating system was configured with the following specifications :

- Memory : 8gb
- Processor : 2 CPUs out of 4
- Virtual Hard Disk : 40gb

2.2.1 Setting up the Oracle VM Virtual Box with Ubuntu OS

1. Download Virtual Box for macOS Mojave
<https://download.virtualbox.org/virtualbox/6.0.14/VirtualBox-6.0.14-133895-OSX.dmg> .
2. Open the downloaded .dmg file . (Figure 3)
3. Allow the installer to complete the installation and launch the virtualbox once it is installed. (Figure 4)





2.2.2 Hadoop Installation

Once Ubuntu is setup in virtualbox, the next step is to install Hadoop. Open terminal by pressing Ctrl + Alt + t once you are inside the ubuntu os.

1. Update the repository using the command

```
(base) hduser@sreenand-VirtualBox:~$ sudo apt-get update
[sudo] password for hduser:
Hit:1 http://ie.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://ie.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
```

2. Install Java and check where java is installed this is necessary for configuring Hadoop.

```
(base) hduser@sreenand-VirtualBox:~$ sudo apt install openjdk-8-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done

(base) hduser@sreenand-VirtualBox:~$ sudo update-alternatives --config java
There is only one alternative in link group java (providing /usr/bin/java): /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java
```

3. Set the java path such that the Hadoop will be able to access the java while running.

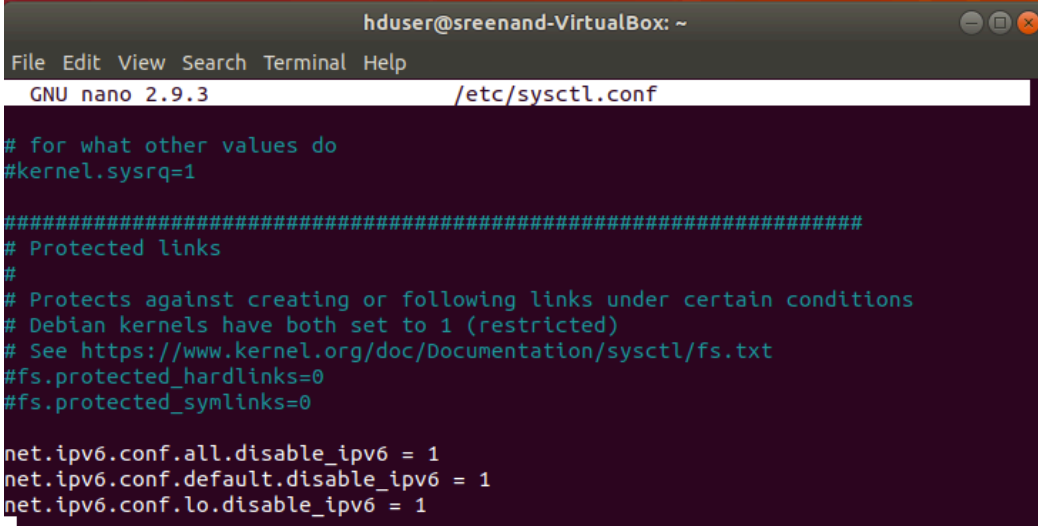
```
hduser@sreenand-VirtualBox: ~
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/profile Modified

for i in /etc/profile.d/*.sh; do
    if [ -r $i ]; then
        . $i
    fi
done
unset i
fi

export JAVA_HOME=/usr
```

- Now source the config file we just edited now, and open the sysctl.conf and add the following commands .

```
(base) hduser@sreenand-VirtualBox:~$ source /etc/profile
(base) hduser@sreenand-VirtualBox:~$ sudo nano /etc/sysctl.conf
```



```
# for what other values do
#kernel.sysrq=1

#####
# Protected links
#
# Protects against creating or following links under certain conditions
# Debian kernels have both set to 1 (restricted)
# See https://www.kernel.org/doc/Documentation/sysctl/fs.txt
#fs.protected_hardlinks=0
#fs.protected_symlinks=0

net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

- Reboot the ubuntu OS.
- To configure the SSH for Hadoop create a user and group with the following names.

```
(base) hduser@sreenand-VirtualBox:~$ sudo addgroup hadoopgroup
(base) hduser@sreenand-VirtualBox:~$ sudo adduser -ingroup hadoopgroup hduser
```

- Install, enable and start ssh and change to hduser

```
(base) hduser@sreenand-VirtualBox:~$ sudo apt-get install ssh
(base) hduser@sreenand-VirtualBox:~$ sudo systemctl enable ssh
(base) hduser@sreenand-VirtualBox:~$ sudo systemctl start ssh
@sreenand-VirtualBox:~$ su - hduser
```

- Generate and authorize all the keys and change the permission to access the keys.

```
(base) hduser@sreenand-VirtualBox:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
(base) hduser@sreenand-VirtualBox:~$ $cat /home/hduser/.ssh/id_rsa.pub >> /home/hduser/.ssh/authorized_keys
(base) hduser@sreenand-VirtualBox:~$ cd .ssh/
(base) hduser@sreenand-VirtualBox:~$ chmod 600 ./authorized_keys
```

- Download and extract Hadoop 2.7.6 using wget :
<https://archive.apache.org/dist/hadoop/core/hadoop-2.7.6/hadoop-2.7.6-src.tar.gz> .

```
(base) hduser@sreenand-VirtualBox:~$ wget https://archive.apache.org/dist/hadoop/core/hadoop-2.7.6/hadoop-2.7.6-src.tar.gz
(base) hduser@sreenand-VirtualBox:~$ tar -xvf hadoop-2.9.2.tar.gz
```

10. Move the extracted files into `cd/usr/local` and create a symbolic link with file name Hadoop.

```
(base) hduser@sreenand-VirtualBox:~$ sudo ln -sf /usr/local/hadoop-2.9.2/ /usr/local/hadoop
```

11. Change the permission for the linked files as we did in step 8 and open the `bashrc` using the command `sudo gedit ~/.bashrc` and configure it. After saving the file, source it.

```
hduser@sreenand-VirtualBox: ~
File Edit View Search Terminal Help
GNU nano 2.9.3 ~/.bashrc

# Hadoop config
export HADOOP_PREFIX=/usr/local/hadoop
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_MAPRED_HOME=${HADOOP_HOME}
export HADOOP_COMMON_HOME=${HADOOP_HOME}
export HADOOP_HDFS_HOME=${HADOOP_HOME}
export YARN_HOME=${HADOOP_HOME}
export HADOOP_CONF_DIR=${HADOOP_HOME}/etc/hadoop
# Native path
export HADOOP_COMMON_LIB_NATIVE_DIR=${HADOOP_PREFIX}/lib/native
export HADOOP_OPTS="-Djava.library.path=${HADOOP_PREFIX}/lib/native"
# Java path
export JAVA_HOME="/usr"
# OS path
export PATH=$PATH:$HADOOP_HOME/bin:$JAVA_HOME/bin:$HADOOP_HOME/sbin
```

12. Open another config file and change the java as shown below.

```
(base) hduser@sreenand-VirtualBox:~$ nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh

# The only required environment variable is JAVA_HOME. All others are
# optional. When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
export JAVA_HOME="/usr"

# The jsvc implementation to use. Jsvc is required to run secure datanodes
```

13. Go to the following directory and append/modify the xml files.

```
(base) hduser@sreenand-VirtualBox:/usr/local/hadoop/etc$ cd /usr/local/hadoop/etc/hadoop
(base) hduser@sreenand-VirtualBox:/usr/local/hadoop/etc/hadoop$ sudo nano core-site.xml

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:9000</value>
(base) hduser@sreenand-VirtualBox:/usr/local/hadoop/etc/hadoop$ sudo nano mapred-site.xml
```

```
hduser@sreenand-VirtualBox: /usr/local/hadoop/etc/hadoop
File Edit View Search Terminal Help
GNU nano 2.9.3 mapred-site.xml

<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
</configuration>
```

```
(base) hduser@sreenand-VirtualBox: /usr/local/hadoop/etc/hadoop$ sudo nano yarn-site.xml
```

```
hduser@sreenand-VirtualBox: /usr/local/hadoop/etc/hadoop
File Edit View Search Terminal Help
GNU nano 2.9.3 yarn-site.xml

<?xml version="1.0"?>
<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<configuration>

<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>

</configuration>
```



```
(base) hduser@sreenand-VirtualBox:/usr/local/hadoop/etc/hadoop$ sudo nano hdfs-site.xml
```

```
<!-- Put site-specific property overrides in this file. -->

<configuration>
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.name.dir</name>
<value>file:/usr/local/hadoop/hadoopdata/hdfs/namenode</value>
[ Read 35 lines ]
^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify
^X Exit          ^R Read File    ^\ Replace      ^U Uncut Text   ^T To Spell
```

14. Now go to the home folder using the command `cd ../../` and then enter into `cd bin` and format the namenode.

```
(base) hduser@sreenand-VirtualBox:/usr/local/hadoop/bin$ sudo hdfs namenode -fo
rmat
```

15. Now start the Hadoop services by typing `start-dfs.sh` and yarn services by using the command `start-yarn.sh`. Once it is done. Type `jps` to see which all process are running, if the output is like the one below, then Hadoop is up and running.

```
(base) hduser@sreenand-VirtualBox:/usr/local/hadoop/bin$ jps
5251 NameNode
5670 SecondaryNameNode
5414 DataNode
5847 ResourceManager
6009 NodeManager
6415 Jps
```

2.2.3 MySQL Installation

1. Install MySQL using the following commands and set password when asked for.

```
(base) hduser@sreenand-VirtualBox:/usr/local/hadoop/bin$ sudo apt-get install m
ysql-server
```

```
(base) hduser@sreenand-VirtualBox:/usr/local/hadoop/bin$ sudo apt-get install m
ysql-client
```

2. Start MySQL

```
(base) hduser@sreenand-VirtualBox:/usr/local/hadoop/bin$ mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.28-0ubuntu0.18.04.4 (Ubuntu)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

2.2.4 Sqoop Installation

1. Download and extract sqoop in cd/usr/local
<http://ftp.heanet.ie/mirrors/www.apache.org/dist/sqoop/1.4.7/sqoop-1.4.7.bin-hadoop-2.6.0.tar.gz> .
2. Assign sqoop directory to Hadoop group and hduser owner.

```
(base) hduser@sreenand-VirtualBox:/usr/local$ sudo chown -R hduser:hadoopgroup sqoop
```

3. Copy or move the mysqlconnectorjavax.x.xx.jar file /usr/share/java to /usr/local/sqoop/lib.

```
(base) hduser@sreenand-VirtualBox:/usr/local$ cp /usr/share/java/mysql-connector-java-5.1.38.jar /usr/local/sqoop/lib/
```

4. Open mapred-site.xml cd/usr/local/Hadoop/etc/Hadoop and add the following.

```
<configuration>
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>

</configuration>
```

5. Open yarn-site.xml cd/usr/local/hadoop/etc/hadoop and add the following.

```
<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>mapreduce.nodemanager.aux-services.mapreduce_shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
</configuration>
```

6. Configure .bashrc file and add the following and source the file.

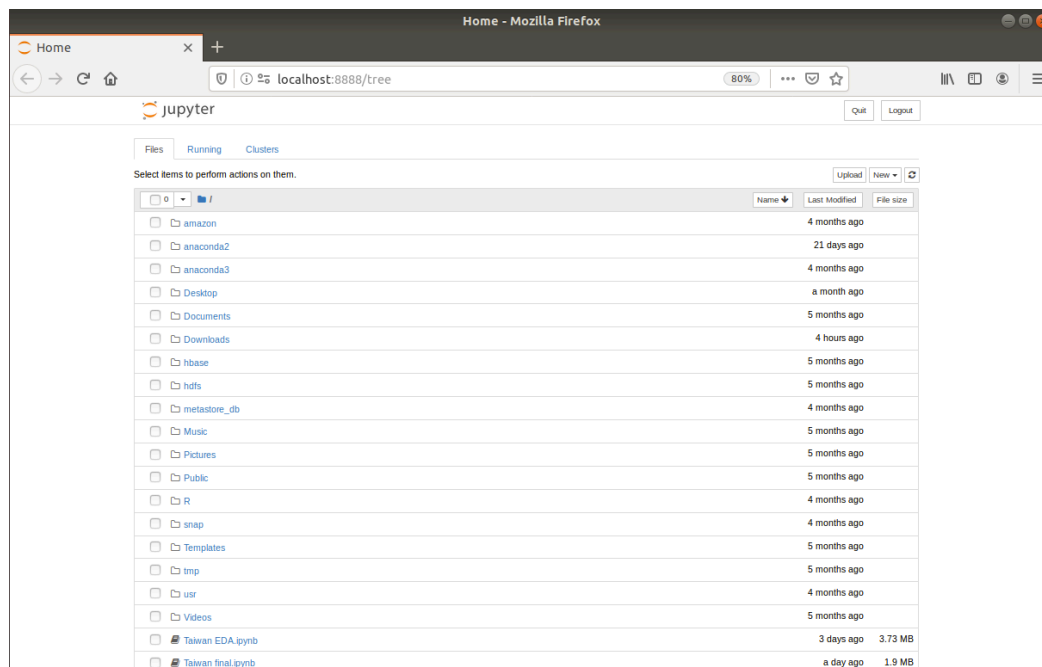
```
# Configure Sqoop environment
export SQOOP_HOME=/usr/local/sqoop
export PATH=$PATH:$SQOOP_HOME/bin
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
```

2.2.5 Apache Spark Installation

1. Download and extract Apache Spark 2.4.3 in cd/usr/local <https://www-eu.apache.org/dist/spark/spark-2.4.3/spark-2.4.3-bin-hadoop2.7.tgz>
2. Change ownership of the spark directory to hadoopgroup and ownership to hduser as seen in section 2.2.4.
3. Configure the .bashrc file as shown below and source the .bashrc.

```
export SPARK_HOME=/usr/local/spark
export PATH=$PATH:$SPARK_HOME/bin
export PYSARK_PYTHON=/usr/local/anaconda2/bin/python2
export PYSARK_DRIVER_PYTHON=jupyter
export PYSARK_DRIVER_PYTHON_OPTS="notebook"
```

4. Type in pyspark in command line and Jupyter notebook will be opened in Firefox browser.



3 Implementation Stage

3.1 Implementation in Apache spark client mode

3.1.1 Importing the required libraries

```
import sys
import pandas as pd
import seaborn as sns
import datetime
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.feature_selection import RFE
from sklearn.linear_model import LassoCV, Lasso, ElasticNet
```

<i>Libraries</i>	<i>Purpose</i>
pandas as pd	Data manipulation and analysis
seaborn as sns	To make statistical graphs
datetime	To manipulate timestamp
numpy as np	Manipulating data types
matplotlib	Plotting of graphs
statsmodels.api as sm	Data exploration using statistics
sklearn.model_selection train_test_split	To split data into train and test dataset
sklearn.linear_model LinearRegression	To perform linear regression
sklearn.neural_network MLPRegressor	To perform neural network regression
sklearn.ensemble RandomForestRegressor	To perform random forest regression
sklearn.ensemble ExtraTreesRegressor	To perform extratrees regression
sklearn.tree DecisionTreeRegressor	To perform decision tree regression
sklearn.ensemble AdaBoostRegressor	To perform AdaBoost regression
sklearn.metrics mean_squared_error, mean_absolute_error, r2_score	To calculate the regression results
sklearn.feature_selection RFE	Feature selection
sklearn.linear_model LassoCV, Lasso, ElasticNet	To perform regularized regression.

3.1.2 Importing the downloaded dataset

1. Download the dataset from : <https://www.kaggle.com/nelsonchu/air-quality-in-northern-taiwan>
2. using the following command shown below (%%time is to find out the time taken for the command to execute).

```
In [4]: %%time
taiwan = pd.read_csv('Taiwan_AQI.csv', parse_dates=True)

<string>:2: DtypeWarning: Columns (10,14,18) have mixed types. Specify dtype option on import or set low_memory=False.

CPU times: user 1.75 s, sys: 274 ms, total: 2.03 s
Wall time: 2.33 s
```

3.1.3 Data Pre-processing and Exploratory Analysis

1. Finding out the data type of each columns.

```
In [3]: taiwan.dtypes

Out[3]: time          object
        station       object
        AMB_TEMP      object
        CH4           object
        CO            object
        NMHC          object
        NO            object
        NO2           object
        NOx           object
        O3            object
        PH_RAIN       object
        PM10          object
        PM2.5         object
        RAINFALL      object
        RAIN_COND     object
        RH            object
        SO2           object
        THC           object
        UVB           object
        UV           object
```

2. Removing unwanted characters from the column name and values.

```
In [5]: taiwan['PM2.5'] = taiwan['PM2.5']

In [6]: taiwan['PM2.5'] = taiwan.PM2.5.str.replace("x", "")
```

3. Converting the datatype from object to float64, because statistical analysis cannot be performed with features having object data type. pd.numeric is used for this purpose.

```
In [7]: %%time
taiwan['AMB_TEMP'] = pd.to_numeric(taiwan.AMB_TEMP, errors='coerce')
taiwan['CO'] = pd.to_numeric(taiwan.NMHC, errors='coerce')
taiwan['CH4'] = pd.to_numeric(taiwan.CH4, errors='coerce')
taiwan['NMHC'] = pd.to_numeric(taiwan.NMHC, errors='coerce')
taiwan['NO'] = pd.to_numeric(taiwan.NO, errors='coerce')
taiwan['NO2'] = pd.to_numeric(taiwan.NO2, errors='coerce')
taiwan['NOx'] = pd.to_numeric(taiwan.NOx, errors='coerce')
taiwan['O3'] = pd.to_numeric(taiwan.O3, errors='coerce')
taiwan['PM10'] = pd.to_numeric(taiwan.PM10, errors='coerce')
taiwan['RH'] = pd.to_numeric(taiwan.RH, errors='coerce')
taiwan['SO2'] = pd.to_numeric(taiwan.SO2, errors='coerce')
taiwan['THC'] = pd.to_numeric(taiwan.TH, errors='coerce')
taiwan['UVB'] = pd.to_numeric(taiwan.UVB, errors='coerce')
taiwan['WD_HR'] = pd.to_numeric(taiwan.WD_HR, errors='coerce')
taiwan['WIND DIREC'] = pd.to_numeric(taiwan.WIND DIREC, errors='coerce')
taiwan['WIND SPEED'] = pd.to_numeric(taiwan.WIND SPEED, errors='coerce')
taiwan['WS_HR'] = pd.to_numeric(taiwan.WS_HR, errors='coerce')
taiwan['PM2.5'] = pd.to_numeric(taiwan.PM2.5, errors='coerce')

CPU times: user 7.32 s, sys: 732 ms, total: 8.05 s
Wall time: 9 s
```

```

In [8]: taiwan.dtypes
Out[8]: time           object
        station        object
        AMB_TEMP       float64
        CH4            float64
        CO             float64
        NMHC           float64
        NO             float64
        NO2            float64
        NOx            float64
        O3            float64
        PH_RAIN        object
        PM10           float64
        PM2.5          object
        RAINFALL        object
        RAIN_COND      object
        RH             float64
        SO2            float64
        THC            float64
        UVB            float64
        WD_HR          float64
        WIND_DIRECT    float64
        WIND_SPEED     float64
        WS_HR          float64
        PM2_5          float64
        dtype: object

```

4. Checking, imputing and removal for NA values.

```

In [38]: data['WS_HR'].fillna(value=data['WS_HR'].median(),inplace=True)

In [39]: for col in ['NO2','NO','NOx','PM10','CO','O3','AMB_TEMP','SO2','WD_HR','RH','WIND_DIRECT','WIND_SPEED','PM2.5']:
        data[col]=data[col].apply(numeric)
        data[col].fillna(value=data[col].median(),inplace=True)
data['RAINFALL'] = data['RAINFALL'].apply(lambda x:0 if x=='NR' else x).apply(numeric)

```

5. Removing unwanted columns and converting all the features into same unit.

```

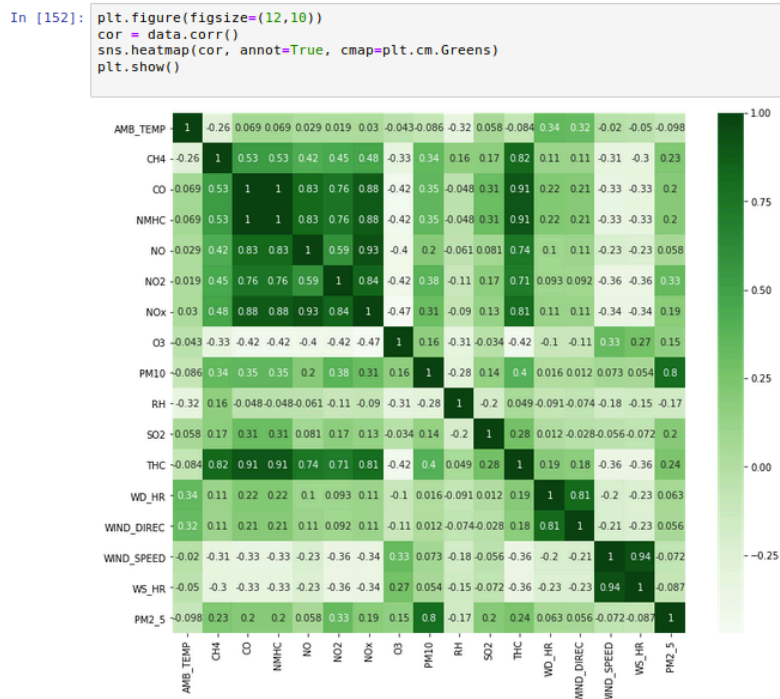
In [8]: taiwan = taiwan.drop(['PM2.5','PH_RAIN','RAIN_COND','UVB'], axis=1)

In [9]: %%time
taiwan['O3'] = taiwan['O3']/1000
taiwan['SO2'] = taiwan['SO2']/1000
taiwan['NO'] = taiwan['NO']/1000
taiwan['NOx'] = taiwan['NOx']/1000
taiwan['NO2'] = taiwan['NO2']/1000

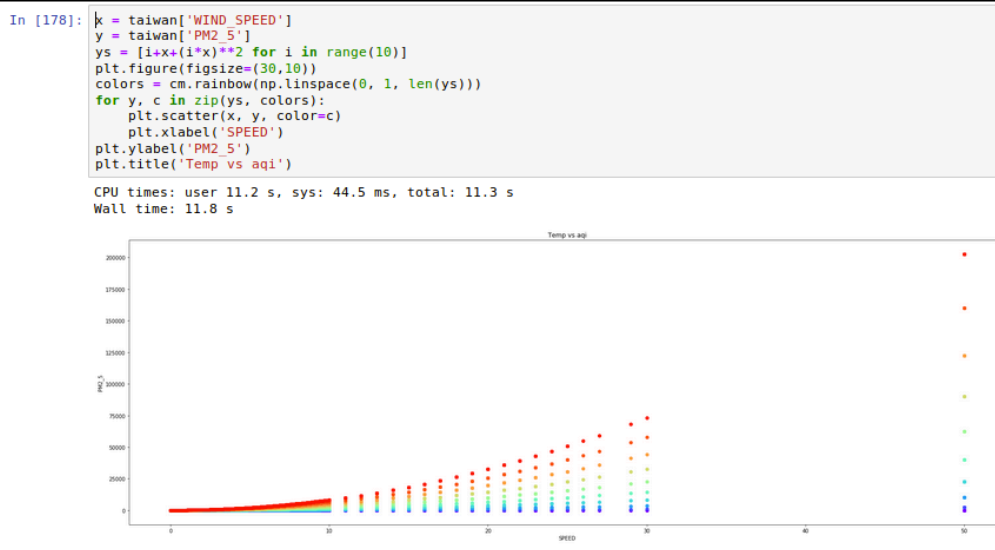
CPU times: user 60.5 ms, sys: 141 ms, total: 201 ms
Wall time: 855 ms

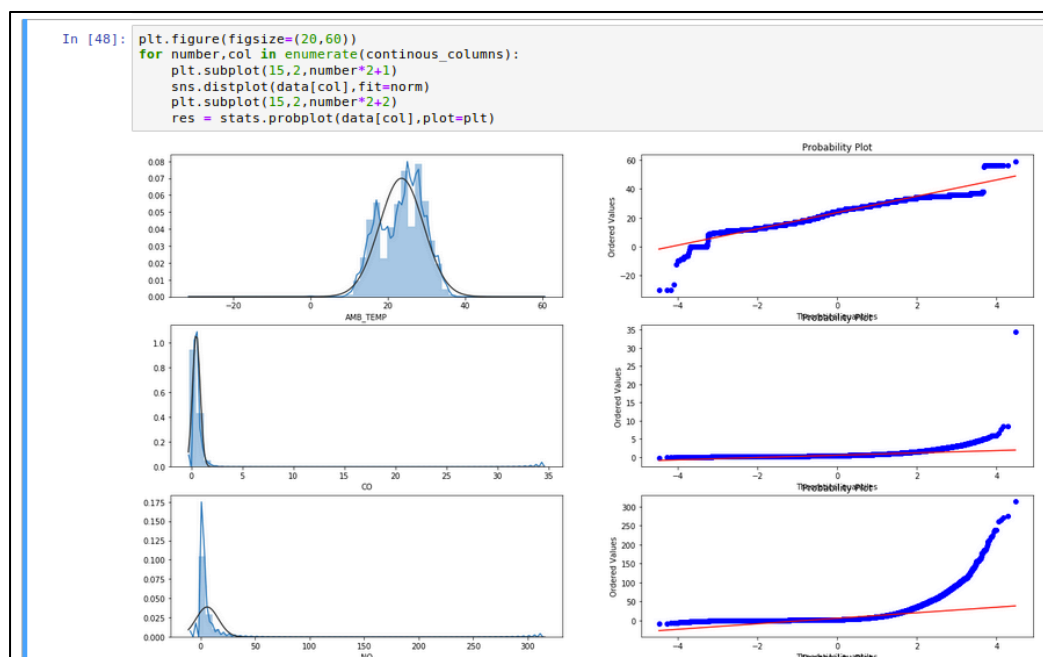
```

6. Checking correlation between the features using correlation matrix.



7. Various exploratory analysis





8. Converting features into cyclic structure.

```

In [187]: %%time
import numpy as np
taiwan['hr_sin'] = np.sin(taiwan.hour*(2.*np.pi/24))
taiwan['hr_cos'] = np.cos(taiwan.hour*(2.*np.pi/24))
taiwan['month_sin'] = np.sin((taiwan.month-1)*(2.*np.pi/12))
taiwan['month_cos'] = np.cos((taiwan.month-1)*(2.*np.pi/12))

CPU times: user 22.5 ms, sys: 439 µs, total: 23 ms
Wall time: 44.8 ms

```

3.1.4 Linear Regression

1. Splitting data into test and train.

```

In [195]: y = df['PM2_5']

In [196]: X = df.drop(['PM2_5','RAINFALL','AQI'], axis=1)

In [197]: %%time
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.3, random_state=1234)

CPU times: user 91.3 ms, sys: 12.3 ms, total: 104 ms
Wall time: 105 ms

```


2. Implementing the model

```
In [200]: %%time
regr = LinearRegression()

CPU times: user 22 µs, sys: 2 µs, total: 24 µs
Wall time: 35 µs

In [ ]: cols = df.columns.tolist()
df.to_csv('/usr/local/cleanT.csv')

In [201]: %%time
A = regr.fit(X_train, y_train)

CPU times: user 233 ms, sys: 37.5 ms, total: 271 ms
Wall time: 292 ms

In [202]: %%time
lin_pred = regr.predict(X_test)

CPU times: user 68.6 ms, sys: 3.89 ms, total: 72.5 ms
Wall time: 75.3 ms

In [203]: linear_regression_score = regr.score(X_test, y_test)
linear_regression_score

Out[203]: 0.6952418147891132
```

3.1.5 Neural Network Regression

```
In [207]: %%time
mlp.fit(X_train, y_train)

CPU times: user 1min 54s, sys: 1min 5s, total: 2min 59s
Wall time: 1min 31s

Out[207]: MLPRegressor(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
beta_2=0.999, early_stopping=False, epsilon=1e-08,
hidden_layer_sizes=(100,), learning_rate='constant',
learning_rate_init=0.001, max_iter=200, momentum=0.9,
n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
random_state=None, shuffle=True, solver='adam', tol=0.0001,
validation_fraction=0.1, verbose=False, warm_start=False)

In [208]: neural_network_regression_score = mlp.score(X_test, y_test)
neural_network_regression_score

Out[208]: 0.7365056592335062

In [209]: nnr_pred = mlp.predict(X_test)
```

3.1.6 Lasso Regression

Lasso regression was found to be not a good fit model, so this model was excluded from the project.

```
In [213]: lasso.fit(X_train, y_train)

Out[213]: Lasso(alpha=1.0, copy_X=True, fit_intercept=True, max_iter=1000,
normalize=False, positive=False, precompute=False, random_state=None,
selection='cyclic', tol=0.0001, warm_start=False)

In [214]: lasso_score = lasso.score(X_test, y_test)
lasso_score

Out[214]: 0.6680752212905757

In [215]: lasso_pred = lasso.predict(X_test)
```

3.1.7 ElasticNet Regression

```
In [219]: elasticnet.fit(X_train, y_train)

Out[219]: ElasticNet(alpha=1.0, copy_X=True, fit_intercept=True, l1_ratio=0.5,
max_iter=1000, normalize=False, positive=False, precompute=False,
random_state=None, selection='cyclic', tol=0.0001, warm_start=False)

In [220]: elasticnet_score = elasticnet.score(X_test, y_test)
elasticnet_score

Out[220]: 0.6718937455784801

In [221]: elasticnet_pred = elasticnet.predict(X_test)
```

3.1.8 Random Forest

```
regr_rf = RandomForestRegressor(n_estimators=300, random_state=1354)

%%time
regr_rf.fit(X_train, y_train)

CPU times: user 5min 47s, sys: 3 s, total: 5min 50s
Wall time: 5min 52s

RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                        max_features='auto', max_leaf_nodes=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=300, n_jobs=None,
                        oob_score=False, random_state=1354, verbose=0, warm_start=False)

decision_forest_score = regr_rf.score(X_test, y_test)
decision_forest_score

0.8026777297339064

regr_rf_pred = regr_rf.predict(X_test)
```

3.1.9 ExtraTreesRegressor

```
extra_tree = ExtraTreesRegressor(n_estimators=200, random_state=2234)

In [232]: %%time
          extra_tree.fit(X_train, y_train)

          CPU times: user 3min 44s, sys: 5.99 s, total: 3min 50s
          Wall time: 3min 52s

Out[232]: ExtraTreesRegressor(bootstrap=False, criterion='mse', max_depth=None,
                               max_features='auto', max_leaf_nodes=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=200, n_jobs=None,
                               oob_score=False, random_state=2234, verbose=0, warm_start=False)

In [233]: extratree_score = extra_tree.score(X_test, y_test)
          extratree_score

Out[233]: 0.8142500347903348

In [234]: %%time
          extratree_pred = extra_tree.predict(X_test)

          CPU times: user 3.59 s, sys: 157 ms, total: 3.75 s
          Wall time: 4.2 s
```

3.1.10 Decision Tree and AdaBoost

```
tree_1 = DecisionTreeRegressor()
tree_2 = AdaBoostRegressor(DecisionTreeRegressor(), n_estimators=200, learning_rate=.5)

[239]: %%time
       tree_1.fit(X_train, y_train)
       tree_2.fit(X_train, y_train)

       CPU times: user 5min 44s, sys: 1.82 s, total: 5min 46s
       Wall time: 5min 53s

[240]: tree_1.score(X_test, y_test)

[240]: 0.5733683402796413

[241]: boosted_tree_score = tree_2.score(X_test, y_test)
       boosted_tree_score

[241]: 0.815636849611419

[242]: %%time
       tree_1_pred = tree_1.predict(X_test)
       tree_2_pred = tree_2.predict(X_test)

       CPU times: user 3.66 s, sys: 1.17 s, total: 4.83 s
       Wall time: 5.76 s
```

3.1.11 XGBoost

XGBoost performance was not up to the mark so this model was not implemented in the research.

```

In [253]: %%time
xboost.fit(xtrain, y_train)

/home/hduser/anaconda2/lib/python2.7/site-packages/xgboost/core.py:587: FutureWarning: Series.base is deprecated and will be removed in a future version
if getattr(data, 'base', None) is not None and \

CPU times: user 9.48 s, sys: 157 ms, total: 9.64 s
Wall time: 13.2 s

Out[253]: XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
colsample_bytree=1, gamma=0, importance_type='gain',
learning_rate=0.1, max_delta_step=0, max_depth=3,
min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
nthread=None, objective='reg:linear', random_state=0, reg_alpha=0,
reg_lambda=1, scale_pos_weight=1, seed=None, silent=True,
subsample=1)

In [252]: xtrain = X_train.drop(['O3_aqi', 'NO2_aqi', 'PM2.5_aqi', 'SO2_aqi', 'CO_aqi', 'PM10_aqi'], axis=1)
xtest = X_test.drop(['O3_aqi', 'NO2_aqi', 'PM2.5_aqi', 'SO2_aqi', 'CO_aqi', 'PM10_aqi'], axis=1)

In [254]: xgb_score = xboost.score(xtest, y_test)
xgb_score

Out[254]: 0.7431200585214719

In [255]: %%time
xboost_pred = xboost.predict(xtest)

CPU times: user 164 ms, sys: 5.04 ms, total: 169 ms
Wall time: 186 ms

```

3.2 Implementation in Apache Spark Cluster Mode

3.2.1 Libraries Used

```

In [1]: from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('lr_example').getOrCreate()
from pyspark.ml import Pipeline
from pyspark.ml.regression import GBRegressor
from pyspark.ml.feature import VectorIndexer
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.regression import LinearRegression
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import RandomForestRegressor

```

<i>Libraries</i>	<i>Purpose</i>
pyspark.sql SparkSession	Pipeline to interact with spark functions
pyspark.ml Pipeline	Provides high level APIs
pyspark.ml.regression GBRegressor	To perform GBT regression
pyspark.ml.feature VectorIndexer	Used to index categorical variables
pyspark.ml.evaluation RegressionEvaluator	Used to evaluate regression results
pyspark.ml.regression LinearRegression	To perform linear regression
pyspark.ml.linalg Vectors	Converts the variables to vectors
pyspark.ml.feature VectorAssembler	Used to combine various vector columns
pyspark.ml.regression RandomForestRegressor	To perform random forest

3.2.2 Importing the csv from HDFS

```

In [4]: %%time
data = spark.read.format("csv").option("header", "true").load('hdfs:///user/cleanT.csv')

CPU times: user 17.9 ms, sys: 893 µs, total: 18.8 ms
Wall time: 20.3 s

```

3.2.3 Changing data type from object to float

```
In [6]: %%time
from pyspark.sql.types import FloatType
data = data.withColumn("CO", data["CO"].cast(FloatType()))
data = data.withColumn("AMB_TEMP", data["AMB_TEMP"].cast(FloatType()))
data = data.withColumn("CH4", data["CH4"].cast(FloatType()))
data = data.withColumn("NMHC", data["NMHC"].cast(FloatType()))
data = data.withColumn("NO", data["NO"].cast(FloatType()))
data = data.withColumn("NO2", data["NO2"].cast(FloatType()))
data = data.withColumn("NOx", data["NOx"].cast(FloatType()))
data = data.withColumn("O3", data["O3"].cast(FloatType()))
data = data.withColumn("PM10", data["PM10"].cast(FloatType()))
data = data.withColumn("PM2_5", data["PM2_5"].cast(FloatType()))
data = data.withColumn("SO2", data["SO2"].cast(FloatType()))
data = data.withColumn("THC", data["THC"].cast(FloatType()))
#data = data.withColumn("UVB", data["UVB"].cast(FloatType()))
data = data.withColumn("WD_HR", data["WD_HR"].cast(FloatType()))
data = data.withColumn("WIND_DIREC", data["WIND_DIREC"].cast(FloatType()))
data = data.withColumn("WIND_SPEED", data["WIND_SPEED"].cast(FloatType()))
data = data.withColumn("WS_HR", data["WS_HR"].cast(FloatType()))
data = data.withColumn("O3_aqi", data["O3_aqi"].cast(FloatType()))
data = data.withColumn("NO2_aqi", data["NO2_aqi"].cast(FloatType()))
data = data.withColumn("RH", data["RH"].cast(FloatType()))
data = data.withColumn("PM2_5_aqi", data["PM2_5_aqi"].cast(FloatType()))
data = data.withColumn("PM10_aqi", data["PM10_aqi"].cast(FloatType()))
data = data.withColumn("SO2_aqi", data["SO2_aqi"].cast(FloatType()))
data = data.withColumn("CO_aqi", data["CO_aqi"].cast(FloatType()))
#data = data.withColumn("aqi", data["aqi"].cast(FloatType()))
data = data.withColumn("month", data["month"].cast(FloatType()))
data = data.withColumn("hour", data["hour"].cast(FloatType()))
data = data.withColumn("day", data["day"].cast(FloatType()))
data = data.withColumn("hr_sin", data["hr_sin"].cast(FloatType()))
data = data.withColumn("hr_cos", data["hr_cos"].cast(FloatType()))
data = data.withColumn("month_sin", data["month_sin"].cast(FloatType()))
data = data.withColumn("month_cos", data["month_cos"].cast(FloatType()))
data = data.withColumn("day_week_cat", data["day_week_cat"].cast(FloatType()))

CPU times: user 171 ms, sys: 72 ms, total: 243 ms
Wall time: 2.44 s
```

3.2.4 Converting the features into vectors

The features are converted into vectors and made into a single column, the column is named as features.

```
In [24]: assembler = VectorAssembler(
    inputCols=["AMB_TEMP",
"CH4",
"CO",
"NMHC",
"NO",
"NO2",
"NOx",
"O3",
"PM10",
"RH",
"SO2",
"THC",
"WD_HR",
"WIND_DIREC",
"WIND_SPEED",
"WS_HR",
"O3_aqi",
"NO2_aqi",
"PM10_aqi",
"SO2_aqi",
"CO_aqi",
"hr_sin",
"hr_cos",
"month_sin",
"month_cos",
"day_week_cat"],
    outputCol="features")

In [32]: output = assembler.transform(data)
```

3.2.5 Splitting the data into train and test

```
In [47]: final_data = output.select("features", "PM2_5")

In [17]: %%time
train_data, test_data = final_data.randomSplit([0.7, 0.3])

CPU times: user 7.76 ms, sys: 636 µs, total: 8.39 ms
Wall time: 116 ms
```

3.2.6 Linear Regression Model

```
In [18]: %%time
         lr = LinearRegression(labelCol='PM2_5')
         CPU times: user 3.24 ms, sys: 1.31 ms, total: 4.56 ms
         Wall time: 240 ms

In [19]: %%time
         lrModel = lr.fit(train_data,)
         CPU times: user 65 ms, sys: 17.5 ms, total: 82.5 ms
         Wall time: 42.2 s

In [21]: %%time
         test_results = lrModel.evaluate(test_data)
         CPU times: user 6.15 ms, sys: 2.45 ms, total: 8.6 ms
         Wall time: 18.7 s

In [22]: unlabeled_data = test_data.select('features')

In [23]: %%time
         predictions = lrModel.transform(unlabeled_data)
         CPU times: user 23.1 ms, sys: 7.99 ms, total: 31.1 ms
         Wall time: 432 ms
```

3.2.7 Random Forest Model

```
In [48]: from pyspark.sql.functions import col
         final_data = final_data.select(col("PM2_5").alias("label"), col("features").alias("features"))
         final_data.show()

+-----+-----+
|label|features|
+-----+-----+
| 78.0|[16.0,2.0999999904...|
| 77.0|[16.0,2.0999999904...|
| 72.0|[16.0,2.0999999904...|
| 65.0|[15.0,2.0,0.11999...|
| 56.0|[15.0,2.0,0.10999...|
| 46.0|[14.0,2.0,0.10999...|
| 45.0|[14.0,2.0,0.14000...|
| 42.0|[14.0,2.0,0.14000...|
| 45.0|[14.0,2.0,0.12999...|
| 46.0|[14.0,2.0,0.14000...|
| 41.0|[15.0,2.0,0.12999...|
| 38.0|[14.0,2.0,0.10999...|
| 36.0|[15.0,2.0,0.10999...|
| 31.0|[14.0,2.0,0.09000...|
| 25.0|[14.0,2.0,0.10000...|
| 25.0|[13.0,2.0,0.10000...|
| 24.0|[13.0,2.0,0.10999...|
| 20.0|[13.0,2.0,0.11999...|
| 22.0|[13.0,2.0,0.12999...|
| 20.0|[13.0,2.0,0.15000...|
+-----+-----+
only showing top 20 rows
```

```
In [49]: featureIndexer = \
         VectorIndexer(inputCol="features", outputCol="indexedFeatures", maxCategories=4).fit(final_data)

In [50]: (trainingData, testData) = final_data.randomSplit([0.7, 0.3])

In [51]: rf = RandomForestRegressor(featuresCol="indexedFeatures")

In [64]: %%time
         pipeline = Pipeline(stages=[featureIndexer, rf])
         CPU times: user 405 µs, sys: 118 µs, total: 523 µs
         Wall time: 538 µs

In [65]: %%time
         model = pipeline.fit(trainingData)
         CPU times: user 49.2 ms, sys: 31.8 ms, total: 81 ms
         Wall time: 20.1 s

In [66]: %%time
         predictions = model.transform(testData)
         CPU times: user 49.2 ms, sys: 9.4 ms, total: 58.6 ms
         Wall time: 461 ms
```

3.2.8 GBT Regressor Model

```
In [82]: gbt = GBTRegressor(featuresCol="indexedFeatures", maxIter=10)

In [83]: pipeline = Pipeline(stages=[featureIndexer, gbt])

In [84]: %%time
model = pipeline.fit(trainingData)
CPU times: user 87.2 ms, sys: 18 ms, total: 105 ms
Wall time: 22.1 s

In [85]: %%time
predictions = model.transform(testData)
CPU times: user 26.3 ms, sys: 2.58 ms, total: 28.9 ms
Wall time: 223 ms
```

4 Loading data into MySQL

1. Table description In MySQL

```
mysql> describe aqi;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| time | varchar(20) | YES | | NULL | |
| station | varchar(20) | YES | | NULL | |
| AMB_TEMP | varchar(20) | YES | | NULL | |
| CH4 | varchar(20) | YES | | NULL | |
| CO | varchar(20) | YES | | NULL | |
| NMHC | varchar(20) | YES | | NULL | |
| NO | varchar(20) | YES | | NULL | |
| NO2 | varchar(20) | YES | | NULL | |
| NOx | varchar(20) | YES | | NULL | |
| O3 | varchar(20) | YES | | NULL | |
| PH_RAIN | varchar(20) | YES | | NULL | |
| PM10 | varchar(20) | YES | | NULL | |
| PM2_5 | varchar(20) | YES | | NULL | |
| RAINFALL | varchar(20) | YES | | NULL | |
| RAIN_COND | varchar(20) | YES | | NULL | |
| RH | varchar(20) | YES | | NULL | |
| SO2 | varchar(20) | YES | | NULL | |
| THC | varchar(20) | YES | | NULL | |
| UVB | varchar(20) | YES | | NULL | |
| WD_HR | varchar(20) | YES | | NULL | |
| WIND_DIRECT | varchar(20) | YES | | NULL | |
| WIND_SPEED | varchar(20) | YES | | NULL | |
| WS_HR | varchar(20) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
```

2. Loading data into the MySQL table

```
mysql> LOAD DATA LOCAL INFILE '/home/hduser/Taiwan_AQI.csv' INTO TABLE aqi FIELDS TERMINATED BY ',' EN
CLOSED BY '"' LINES TERMINATED BY '\n' IGNORE 1 ROWS;
Query OK, 218640 rows affected (11.03 sec)
Records: 218640 Deleted: 0 Skipped: 0 Warnings: 0
```

5 Conclusion

The above steps were performed in building the AQI prediction model for the Island of Taiwan. Various machine learning techniques with the libraries used to deploy them are mentioned above. All the models were implemented in Apache spark environment in both Apache spark client and cluster environment. It is to be noted that there can be few glitches if any other versions of Hadoop or Spark is installed rather than the above mentioned versions.