Configuration Manual

MSc Research Project
Data Analytics

Aniket Bhawkar
Student ID: x17170885

School of Computing
National College of Ireland

Supervisor: Dr. Muhammad Iqbal

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Aniket Bhawkar
Student ID:	x17170885
Programme:	Data Analytics
Year:	2018-19
Module:	MSc Research Project
Supervisor:	Dr. Muhammad Iqbal
Submission Due Date:	12/08/2019
Project Title:	Configuration Manual
Word Count:	835
Page Count:	8

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:	8th August 2019

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
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.	☐

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

Aniket Bhawkar
x17170885

1 Introduction

The configuration manual provides the information about the software and hardware requirements and it setup process. This document involves screenshot and step-by-step guide for executing the research "Recommendation System based on Behaviour Patterns from Social Media Images".

2 Installation

2.1 Anaconda

Link - `https://www.anaconda.com/distribution/`
Label Encoding, Data Formation, Modeling, Extrapolation Testing and Evaluation is done using Python. Hence, to execute the research it is essential to download and install the latest version of Python (In this case consider Anaconda).

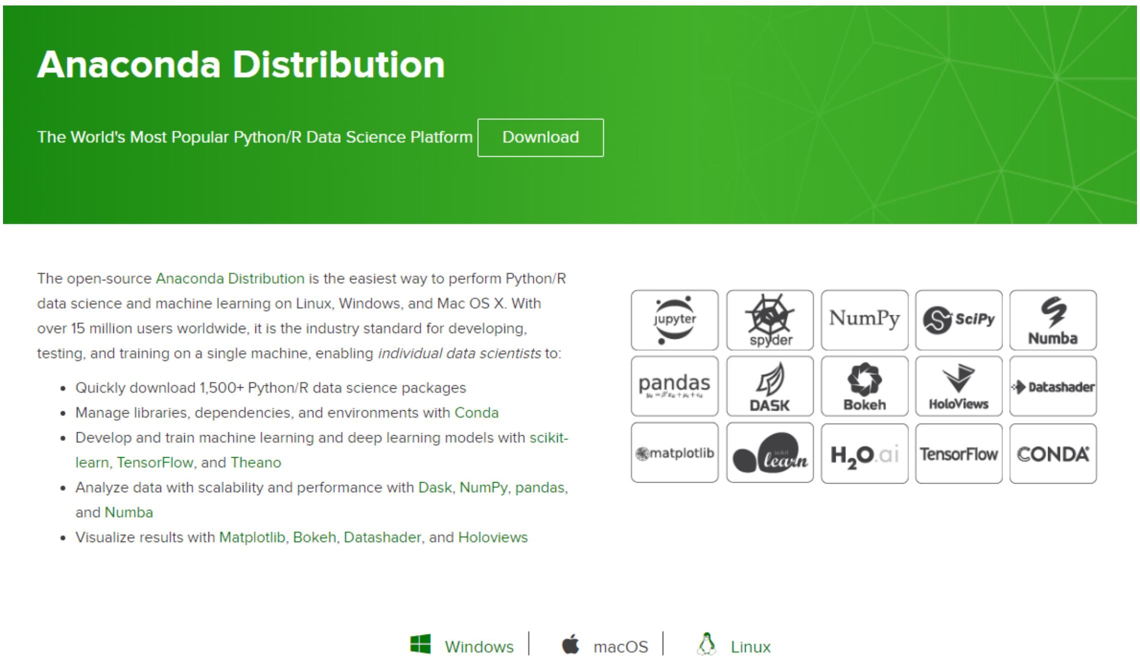


Figure 1: Anaconda Download Webpage

2.2 R Studio

Link - `https://www.rstudio.com/products/rstudio/download/`
Data cleaning, categorisation and data pre-processing is carried out using R Studio. These being a vital step in any project, R Studio is supposed to be installed. Consider downloading and installing the Free version from the above mentioned link.

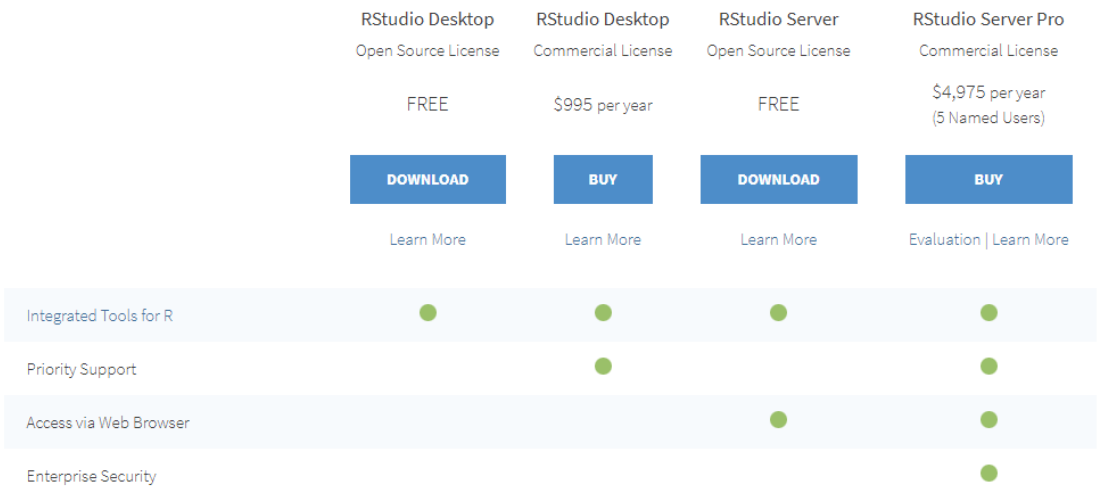


Figure 2: Anaconda Download Webpage

2.3 Jupyter Notebook

Open Anaconda Command Prompt; Click on ***Start*** and search for ***Anaconda***. Then after type the following command

pip install jupyterlab

2.4 Tensorflow and Keras

After executing the above command type the following commands in the command prompt.

pip install tensorflow
pip install keras

2.5 IBM SPSS

IBM SPSS is used for Chi Square Analysis in determining the impact of cultural differences on behavioural patterns related to tourism. Hence, IBM SPSS is supposed to be downloaded and installed from `https://www.ibm.com/products/spss-statistics`.

2.6 Tableau

Tableau specialises in representing the information in graphical format. The results received from the models are visualised using Tableau in order to provide deeper insights. Tableau can be downloaded from `https://www.tableau.com/products/desktop/download`

2.7 System Configuration

- Processor: Intel(R) Core(TM) i5-8250 CPU @ 1.60GHz 1.80GHz
- 64-bit Windows Operating System, x64-based processor
- RAM: 8GB
- Storage: 1TB

3 Server

For this research, a Free Tier instance of Amazon EC2 - `https://aws.amazon.com/ec2/?nc2=h_m1` was considered. The inter-server communication using the RESTful API with the computer vision tool is done using this instance. Amazon Rekognition - `https://aws.amazon.com/rekognition/?nc2=h_m1` is utilised for generating labels from the shared images.

3.1 Instance Configuration

- Ubuntu 18.04 LTS
- 1 CPU, 1 GB RAM
- 30 GB storage
- Port 80, 443, 3306 open for connection

3.2 Apache2

Apache2 can execute the PHP files which are used for making the REST API calls. For this execute the following commands in the terminal.

sudo apt-get update
sudo apt install apache2
sudo systemctl restart apache2

3.3 MySQL

For storing the extracting user information, image information, labels, categories, image filenames, etc; a MySQL database is needed.

sudo apt-get update

sudo apt-get install mysql-server

4 Label Extraction

The labels are extracted from the images using Amazon Rekognition. Figure 3 represents label extraction sample code. The code is executed in a for-loop which enables extracting labels from numerous images in a single execution. Labels above confidence score of 70% are only fetched. The extracted labels are then stored in a MySQL database and are used for further processing.

```
$imgname = $row["imageName"];
$results = $client->detectLabels([
        'Image' => [
                'Bytes' => file_get_contents($log_directory.$row["imageName"]),
        ],
        'MaxLabels' => 10,
        'MinConfidence' => 70,
]);
foreach($results["Labels"] as $l){
        $labelname = $l["Name"];
        $confidence = $l["Confidence"];
        $sql = "INSERT INTO labels (imageName, labelName, confidence) VALUES ('$imgname', '$labelname', '$confidence')";
        if ($conn->query($sql) === TRUE) {
                //echo "New record created successfully";
        }else{
                echo "Error: " . $sql . "<br>" . $conn->error;
        }
}
```

Figure 3: Label Extraction Sample Code (detectLabels.php)

5 Categorisation

These extracted labels are diverse and hence its essential to assign them to a particular category. Categorisation involves label dictionary in the form of array which assigns a main category to the image. Figure 4 highlights the sample code used for this categorisation process.

```
            "Bread", "Bun", "Sweets", "Grass", "Vegetation", "Fruit", "Cherry", "Poultry", "Chicken",
            "Seasoning", "Sandwich", "Orange", "Citrus Fruit", "French Toast", "Toast", "Breakfast",
            "Vegetable", "Stew", "Bowl", "Pizza", "Bean", "Steak", "Ribs", "Cutlery", "Spoon", "Lunch",
            "Cornbread", "Pancake", "Pita", "Glass", "Wine", "Beverage", "Alcohol", "Wine Glass", "Red Wine",
            "Bottle", "Sprout", "Fries", "Lentil", "Fork", "Garlic", "Burrito", "Dessert", "Cake",
            "Birthday Cake", "Dining Table", "Food Court", "Vessel", "Apple", "Produce", "Restaurant",
            "Cafeteria", "Coffee Cup", "Coffee", "Chocolate", "Milk", "Drink", "Cocktail", "Seafood",
            "Fish", "Lobster", "Spaghetti", "Watermelon", "Dinner", "Soda", "Coke", "Beer", "Cola", "Bakery",
            "Banana")

df$Category = ifelse(is.element(df$Label, nature),"Nature",
                    ifelse(is.element(df$Label, adventure),"Adventure",
                           ifelse(is.element(df$Label, beach),"Beach",
                    ifelse(is.element(df$Label, food),"Food",
                           ifelse(is.element(df$Label, architecture),"Architecture",
                                  ifelse(is.element(df$Label, clothing),"Clothing","none"))))))


df$Name = as.factor(df$Name)
write.csv(df, "./dataset.csv",row.names = FALSE)
```

Figure 4: Sample code of Categorisation of Labels (cleaning.R)

6 Modelling

KNN, KMeans, RNN-LSTM and DBSCAN algorithms are being utilised in this experimentation. The code has been developed in Python and the data is imported from a csv file. First, all the necessary libraries are imported.

6.1 KNN

Figure 5 highlights the code which is responsible for providing the output of the KNN algorithm.

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=5, p=2,
           weights='uniform')
```

```
y_pred = classifier.predict(X_test)
```

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[3 0 0 0]
 [3 2 0 0]
 [0 0 2 0]
 [0 0 3 0]]
              precision    recall  f1-score   support

           0       0.50      1.00      0.67         3
           1       1.00      0.40      0.57         5
           2       0.40      1.00      0.57         2
           3       0.00      0.00      0.00         3

   micro avg       0.54      0.54      0.54        13
   macro avg       0.47      0.60      0.45        13
weighted avg       0.56      0.54      0.46        13
```

Figure 5: Sample Code for KNN algorithm (KNN.ipynb)

6.2 KMeans

The sample code of KMeans algorithm is shown in Figure 6. The algorithm is executed on default and tuned up settings.

```
kmeans = KMeans(n_clusters=6)
kmeans.fit(X)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=6, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

```
correct = 0
for i in range(len(X)):
    predict_me = np.array(X[i].astype(float))
    predict_me = predict_me.reshape(-1, len(predict_me))
    prediction = kmeans.predict(predict_me)
    if prediction[0] == y[i]:
        correct += 1
print(correct/len(X))
```

```
0.625
```

```
kmeans =  KMeans(n_clusters=6, n_init=20, max_iter=4000, precompute_distances=False, algorithm='full', copy_x=False)
kmeans.fit(X)
```

```
KMeans(algorithm='full', copy_x=False, init='k-means++', max_iter=4000,
    n_clusters=6, n_init=20, n_jobs=None, precompute_distances=False,
    random_state=None, tol=0.0001, verbose=0)
```

```
correct = 0
for i in range(len(X)):
    predict_me = np.array(X[i].astype(float))
    predict_me = predict_me.reshape(-1, len(predict_me))
    prediction = kmeans.predict(predict_me)
    if prediction[0] == y[i]:
        correct += 1

print(correct/len(X))
```

```
0.625
```

Figure 6: Sample Code for KMeans algorithm (KMeans.ipynb)

6.3 RNN-LSTM

Figure 7 displays the execution of the algorithm. It marks the training of the model for 500 epochs.

```
history = model.fit(x_train,y_train, epochs=500, validation_data=(x_test,y_test))
```

```
Epoch 1/500
85/85 [==============================] - 2s 25ms/step - loss: 0.2070 - acc: 0.1059 - val_loss: 0.1416 - val_acc: 0.1034
Epoch 2/500
85/85 [==============================] - 0s 2ms/step - loss: 0.1640 - acc: 0.1059 - val_loss: 0.1192 - val_acc: 0.1034
Epoch 3/500
85/85 [==============================] - 0s 2ms/step - loss: 0.1411 - acc: 0.1059 - val_loss: 0.1369 - val_acc: 0.1034
Epoch 4/500
85/85 [==============================] - 0s 2ms/step - loss: 0.1359 - acc: 0.1059 - val_loss: 0.1484 - val_acc: 0.1034
Epoch 5/500
85/85 [==============================] - 0s 2ms/step - loss: 0.1386 - acc: 0.1059 - val_loss: 0.1530 - val_acc: 0.1034
Epoch 6/500
85/85 [==============================] - 0s 2ms/step - loss: 0.1380 - acc: 0.1059 - val_loss: 0.1484 - val_acc: 0.1034
Epoch 7/500
85/85 [==============================] - 0s 2ms/step - loss: 0.1369 - acc: 0.1059 - val_loss: 0.1447 - val_acc: 0.1034
Epoch 8/500
85/85 [==============================] - 0s 2ms/step - loss: 0.1359 - acc: 0.1059 - val_loss: 0.1417 - val_acc: 0.1034
Epoch 9/500
85/85 [==============================] - 0s 2ms/step - loss: 0.1359 - acc: 0.1059 - val_loss: 0.1367 - val_acc: 0.1034
Epoch 10/500
```

```
results = model.predict(x_test)
```

```
plt.scatter(range(29), results, c='r')
plt.scatter(range(29), y_test, c='g')
plt.show()
```

Figure 7: Sample Code for RNN-LSTM algorithm (LSTM.ipynb)

6.4 DBSCAN

DBSCAN algorithm is utilised for providing insights about the popular destinations those are preferred. It scans the dataset and highlights the popular geo-locations. Figure 8 shows the code used for implementing this model.

```
kms_per_radian = 6371.0088
epsilon = 1.5 / kms_per_radian
db = DBSCAN(eps=epsilon, min_samples=1, algorithm='ball_tree', metric='haversine').fit(np.radians(coords))
cluster_labels = db.labels_
num_clusters = len(set(cluster_labels))
clusters = pd.Series([coords[cluster_labels == n] for n in range(num_clusters)])
print('Number of clusters: {}'.format(num_clusters))
```

```
Number of clusters: 25
```

```
def get_centermost_point(cluster):
    centroid = (MultiPoint(cluster).centroid.x, MultiPoint(cluster).centroid.y)
    centermost_point = min(cluster, key=lambda point: great_circle(point, centroid).m)
    return tuple(centermost_point)
centermost_points = clusters.map(get_centermost_point)
```

```
lats, lons = zip(*centermost_points)
rep_points = pd.DataFrame({'lon':lons, 'lat':lats})
```

```
rs = rep_points.apply(lambda row: dataset[(dataset['lat']==row['lat']) & (dataset['lon']==row['lon'])].iloc[0], axis=1)
```

```
fig, ax = plt.subplots(figsize=[10, 6])
rs_scatter = ax.scatter(rs['lat'], rs['lon'], c='#99cc99', edgecolor='None', alpha=0.7, s=120)
df_scatter = ax.scatter(dataset['lat'], dataset['lon'], c='k', alpha=0.9, s=3)
ax.set_title('Full data set vs DBSCAN reduced set')
ax.set_xlabel('Latitude')
ax.set_ylabel('Longitude')
ax.legend([df_scatter, rs_scatter], ['Full set', 'Reduced set'], loc='upper right')
plt.show()
```

Figure 8: Sample Code for DBSCAN (DBSCAN.ipynb)

7 Evaluation

The results received from the above mentioned algorithms could be used to evaluate the accuracy. These algorithms are tested over various number of people groups. The results are then extrapolated using linear regression modelling technique. Chi Square Test is done using IBM SPSS to analyse the impact of cultural difference on tourism. DBSCAN algorithm provides clusters which are useful for KDD.

7.1 Linear Regression

Figure 7.2 is the sample code which is used for linear regression for extrapolation testing. The precision values received from KNN algorithm and accuracy from KMeans algorithm are fed as input to this evaluation technique. The output justifies a slope.

```
data = pd.read_csv('F:/Masters/Semester 3/Thesis/x17170885/Output/Extrapolation/KMeans.csv')
X = data.iloc[:, 0].values.reshape(-1, 1)  # values converts it into a numpy array
Y = data.iloc[:, 1].values.reshape(-1, 1)  # -1 means that calculate the dimension of rows, b
linear_regressor = LinearRegression()  # create object for the class
linear_regressor.fit(X, Y)  # perform linear regression
Y_pred = linear_regressor.predict(X)  # make predictions
```

```
plt.scatter(X, Y)
plt.plot(X, Y_pred, color='red')
plt.show()
```

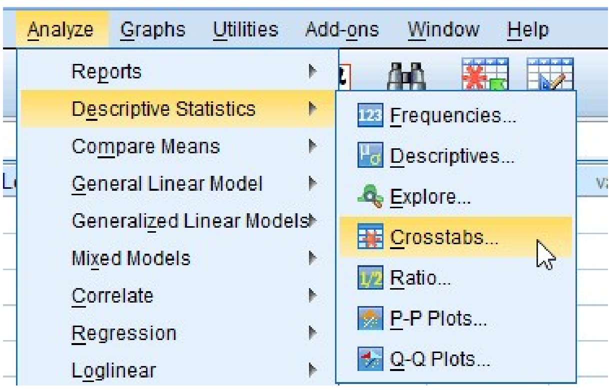
```
data = pd.read_csv('F:/Masters/Semester 3/Thesis/x17170885/Output/Extrapolation/KNN.csv')  #
X = data.iloc[:, 0].values.reshape(-1, 1)  # values converts it into a numpy array
Y = data.iloc[:, 1].values.reshape(-1, 1)  # -1 means that calculate the dimension of rows, b
linear_regressor = LinearRegression()  # create object for the class
linear_regressor.fit(X, Y)  # perform linear regression
Y_pred = linear_regressor.predict(X)  # make predictions
```

```
plt.scatter(X, Y)
plt.plot(X, Y_pred, color='red')
plt.show()
```

Figure 9: Sample code for Linear Regression algorithm for Extrapolation testing (Linear Regression Extrapolation.ipynb)

7.2 Chi-square Analysis

Chi Square test is done using IBM SPSS. This can be done by clicking on
Analyze > Descriptives Statistics > Crosstabs...
Select rows and columns



Click on **Statistics** button and select **Chi-Square**.