National College of Ireland

Configuration Manual

MSc Research Project

Data Analytics

Sadhvi Dubey

Student ID: 19199350

School of Computing

National College of Ireland

Supervisor: Dr. Vladimir Milosavljevic

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Sadhvi Dubey
Student ID:	19199350
Programme:	Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Dr. Vladimir Milosavljevic
Submission Due Date:	16/12/2021
Project Title:	Configuration Manual
Word Count:	504
Page Count:	7

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:	Sadhvi Dubey
Date:	16th December 2021

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

Sadhvi Dubey
19199350

1 Introduction

This document content list of step required execute the code used to build this research project.This project is build using coding language Python and to run the code we have used Anaconda Jupyter Notebook. In this project python version 3.6 is used hence the relevant version is needed. ((Sukumaran and Holder; 2010)).

2 System Requirement

Below section provides the hardware and software requirement to execute the files associated with the project.

2.1 System Hardware Specification

Below are the list of hardware specification of the system used in this research project
Processor: MAC M1 Processor
Storage:1TB HDD
RAM: 16 GB
Operating System(OS): Mac OS

2.2 System Software Specification

In this project we have used following software:
1. Install Anaconda(Jupyter notebook environment)
2. Python version 3.6
3. Microsoft Excel
4. Tableau

3 Environment setup

We need to first Install Anaconda and launch juypter Notebook.

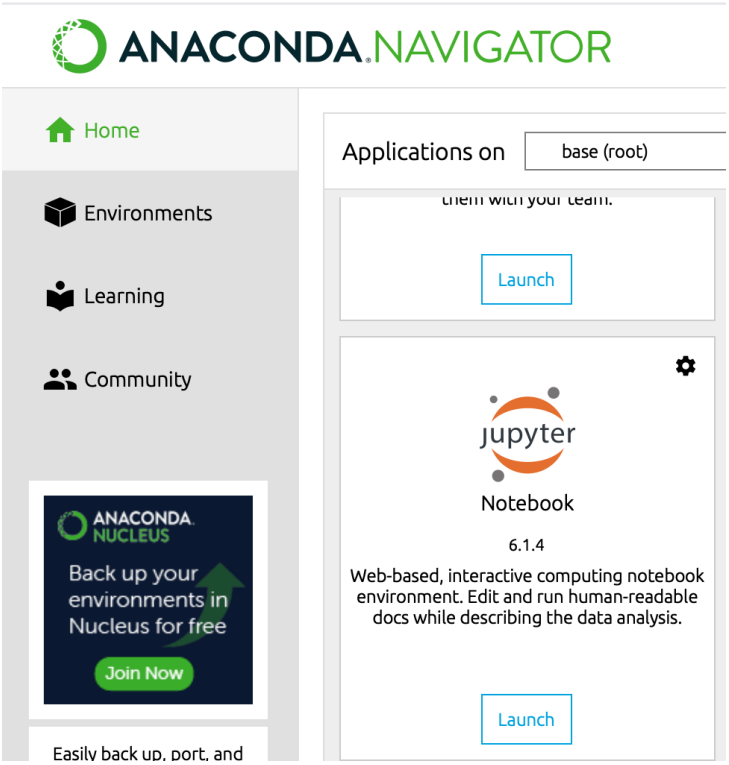

Figure 1: Data Collection

Data Source :

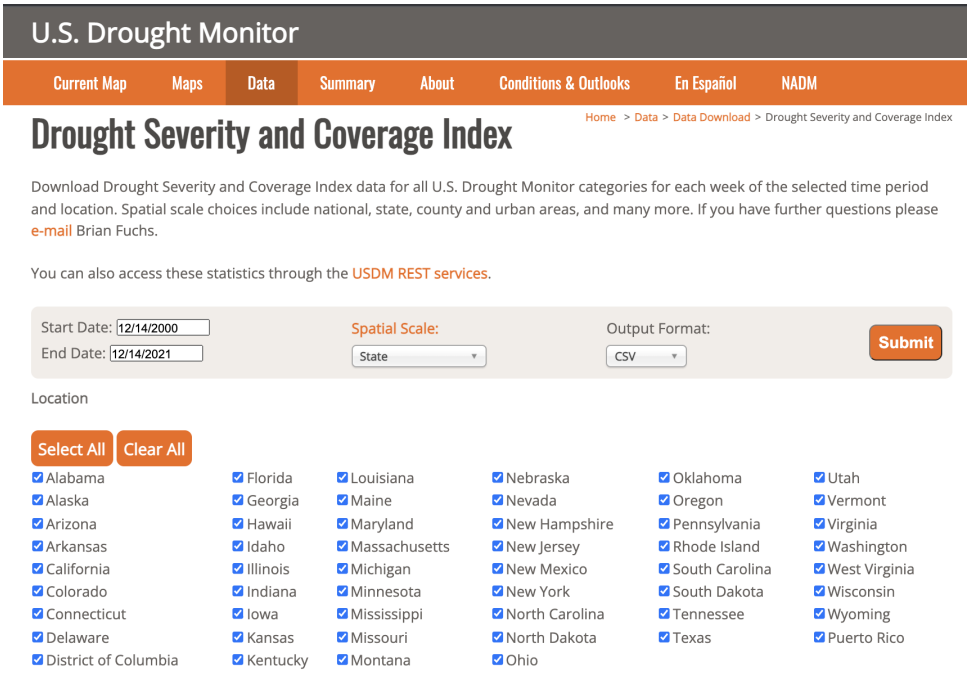

Figure 2: Data Collection

Import Libraries

```
import pandas as pd
import tensorflow as tf
import numpy as np
import keras
from keras.layers import Input, Dense, Embedding, Conv2D, MaxPool2D
from keras.layers import Reshape, Flatten, Dropout, Concatenate
from keras.callbacks import ModelCheckpoint
from keras.optimizers import Adam
from keras.models import Model
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
import datetime
import time
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import scipy.sparse as sp
from scipy.sparse import vstack
from scipy import sparse
from scipy.sparse.linalg import spsolve
from subprocess import check_output
from sklearn import metrics
```

Figure 3: Import Python Libraries

3.1 Data Import and Data Pre-processing

Importing and Cleaning Data

We'll take the polygons of US counties from the bokeh library, and we'll import the dataset from the USDM. Let's look at the USDM data.

```
from bokeh.sampledata.us_counties import data as counties
drought=pd.read_csv("C:/document/Droughts/us-droughts.csv.zip",compression="zip", encoding='latin1')
drought.head()
```

Figure 4: Mount Data and Data Processing

4 Predictive Model

The below piece of code is used to compare the ML Models in the initial analysis in order to define a baseline for the California datasest.

KNeighborsClassifier

```
# Train an KNeighborsClassifier model
# define an KNeighborsClassifier model and fit on xtrain and ytrain
knn_classifier = KNeighborsClassifier()
knn_classifier.fit(xtrain,ytrain)

# predict with the help of trained KNeighborsClassifier model
y_pred_knn = knn_classifier.predict(xtest)
```

```
# view various evaluation matrices
print("accuracy_score:", accuracy_score(ytest,y_pred_knn))
print("f1_score:", f1_score(ytest,y_pred_knn,average = 'micro'))
print(classification_report(ytest,y_pred_knn))
```

```
accuracy_score: 0.8143254155524094
f1_score: 0.8143254155524094
              precision    recall  f1-score   support

           0       0.83      0.85      0.84     10731
           1       0.80      0.77      0.78      8340

    accuracy                           0.81     19071
   macro avg       0.81      0.81      0.81     19071
weighted avg       0.81      0.81      0.81     19071
```

Figure 5: Applied KNN

5 Predictive Model

Comparing ML Model

```
# Train a RandomForestClassifier model
# define a RandomForestClassifier model and fit on xtrain and ytrain
randomf_classifier2 = RandomForestClassifier(n_estimators = 500, random_state = 42)
randomf_classifier2.fit(xtrain,ytrain)

# predict with the help of trained KNeighborsClassifier model
y_pred_randomf2 = randomf_classifier2.predict(xtest)
```

```
# view various evaluation matrices
print("accuracy_score:", accuracy_score(ytest,y_pred_randomf2))
print("f1_score:", f1_score(ytest,y_pred_randomf2,average = 'micro'))
print(classification_report(ytest,y_pred_randomf2))
```

```
accuracy_score: 0.8463111530596193
f1_score: 0.8463111530596193
              precision    recall  f1-score   support

           0       0.84      0.90      0.87     10731
           1       0.85      0.78      0.82      8340

    accuracy                           0.85     19071
   macro avg       0.85      0.84      0.84     19071
weighted avg       0.85      0.85      0.85     19071
```

Figure 6: Applied Random Forest

```
drought_features = X_norm.columns
feature_importances = randomf_classifier2.feature_importances_
indices = np.argsort(feature_importances)[-18:]  # all 18 features
plt.title('Feature Importances')
plt.barh(range(len(indices)), feature_importances[indices], color='r', align='center')
plt.yticks(range(len(indices)), [drought_features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```

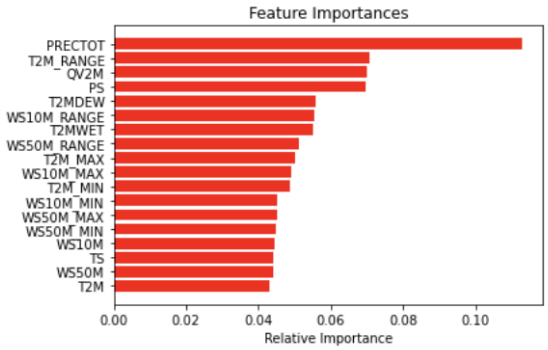

Figure 7: Relative Important Features

We have imported necessary libraries for execution of LSTM with Inception v3 and DenseNet121 as shown below.

Logistic Regression

In [13]:
```
# LogReg_clf = LogisticRegression(random_state = 0, max_iter=15000, solver='saga',penalty='elasticnet',l1_ratio=1) #0.7
# LogReg_clf = LogisticRegression(random_state = 0, max_iter=10000, solver='liblinear',penalty='l1') 0.7112893922709873

# Train a Logistic Regression model
# define an logistic regression model and fit on xtrain and ytrain
LogReg_classifier = LogisticRegression(random_state = 0, max_iter=500, solver='lbfgs',penalty='none') # 0.7124954118819
LogReg_classifier.fit(xtrain, ytrain)

# predict with the help of trained logistic regression model
y_pred_lr = LogReg_classifier.predict(xtest)
```

In [14]:
```
# view various evaluation matrices
print("accuracy_score:", accuracy_score(ytest,y_pred_lr))
print("f1_score:", f1_score(ytest,y_pred_lr,average = 'micro'))
print(classification_report(ytest,y_pred_lr))
```

```
accuracy_score: 0.6820827434324367
f1_score: 0.6820827434324367
              precision    recall  f1-score   support

           0       0.68      0.82      0.74     10731
           1       0.69      0.50      0.58      8340

    accuracy                           0.68     19071
   macro avg       0.68      0.66      0.66     19071
weighted avg       0.68      0.68      0.67     19071
```

Figure 8: Logistic Regression

```
In [ ]: import torch
from torch import nn
from sklearn.metrics import f1_score, mean_absolute_error

class DroughtNetLSTM(nn.Module):
    def __init__(
        self,
        output_size,
        num_input_features,
        hidden_dim,
        n_layers,
        ffnn_layers,
        drop_prob,
        static_dim=0,
    ):
        super(DroughtNetLSTM, self).__init__()
        self.output_size = output_size
        self.n_layers = n_layers
        self.hidden_dim = hidden_dim

        self.lstm = nn.LSTM(
            num_input_features,
            hidden_dim,
            n_layers,
            dropout=drop_prob,
            batch_first=True,
        )
        self.dropout = nn.Dropout(drop_prob)
        self.fflayers = []
        for i in range(ffnn_layers - 1):
            if i == 0:
                self.fflayers.append(nn.Linear(hidden_dim + static_dim, hidden_dim))
            else:
                self.fflayers.append(nn.Linear(hidden_dim, hidden_dim))
        self.fflayers = nn.ModuleList(self.fflayers)
        self.final = nn.Linear(hidden_dim, output_size)

    def forward(self, x, hidden, static=None):
        batch_size = x.size(0)
        x = x.to(dtype=torch.float32)
        if static is not None:
            static = static.to(dtype=torch.float32)
        lstm_out, hidden = self.lstm(x, hidden)
        lstm_out = lstm_out[:, -1, :]

        out = self.dropout(lstm_out)
        for i in range(len(self.fflayers)):
            if i == 0 and static is not None:
                out = self.fflayers[i](torch.cat((out, static), 1))
            else:
                out = self.fflayers[i](out)
        out = self.final(out)

        out = out.view(batch_size, -1)
        return out, hidden

    def init_hidden(self, batch_size):
        weight = next(self.parameters()).data
        hidden = (
            weight.new(self.n_layers, batch_size, self.hidden_dim).zero_().to(device),
            weight.new(self.n_layers, batch_size, self.hidden_dim).zero_().to(device),
        )
        return hidden
```

Figure 9: Applied LSTM

5.1 Plotting Map

Using below code we have plotted map, To run this code it might take 5-7min. The color palette changes from blue to red which makes visualization clear to understand the drought affected region.

```
In [117]: #1
counties_drought_continental=counties_drought[~counties_drought["state"].isin(['hi', 'ha','ak','pr','gu','mp','as','vi
#2
counties_drought_dict=counties_drought_continental.to_dict("index")
counties_drought_list = [dict(county)for cid, county in counties_drought_dict.items()]
#3
dates_list=[]
for year in range(2010,2020):
    for month in range(1,13):
        if month<10:
            month="0"+str(month)
        date=str(year)+"-"+str(month)
        dates_list.append(date)
```

Now we plot the thing! If you run this yourself it might take 20 seconds. The rainbow color palette goes from blue to red which makes it a nice visualization tool for drought. We'll plot January, 2010 first.

```
In [118]: cbar_tick_labels=[(0,'Not in drought'),(1,'Abnormally dry conditions'),(2,'Moderate Drought'),(3,'Severe Drought'),(4,
polygons = hv.Polygons(counties_drought_list, ['lons', 'lats'], vdims=hv.Dimension(dates_list[1], range=(0, 5)), label=
polygons.options(logz=False, xaxis=None, yaxis=None, cbar_ticks=cbar_tick_labels,
                 show_grid=False, show_frame=False, colorbar=True, color_index=dates_list[1],
                 fig_size=500, edgecolor='black', cmap=list(rainbow))
```

Figure 10: US continent

References

Sukumaran, J. and Holder, M. T. (2010). Dendropy: a python library for phylogenetic computing, *Bioinformatics* **26**(12): 1569–1571.