PM2.5 & PM10 Prediction using ANN

Akrit Woranithiphong 6358104056

February 25, 2022

1 Abstract

This assignment is based on predicting Bangkok's air pollution; PM2.5 & PM10 using Artificial Neural Network (ANN). In this assignment, we will be going through the process of data preparation up to creating a ANN using Scikit-Learn.

2 Materials and Methods

2.1 Materials

The codebase will be coded in Python for simplicity as expected for the assignment. The dataset we will be using is a Bangkok Air Pollution dataset derived from https://aqicn.org/data-platform. We will be using Scikit-learn, Pandas, Numpy, and Matplotlib Python Library for Artificial Neural Network Algorithm, Data Manipulation, Data Structure, Plotting/Visualization, respectively.

2.2 Methods

2.2.1 Getting Started

In the first process, we will be downloading IDE, which in this case is Anaconda for Jupyter Notebook, and importing Scikit-learn, Pandas, Numpy, and Matplotlib into our Python's virtual environment.

2.2.2 Data Preperation

2.2.2.1 Dealing with Missing Value

Second, we will be dropping missing PM2.5 & PM10 missing rows. The reason behind not filling these missing rows is that both PM2.5 & PM10 is our output/prediction value and by filling it up with unrealistic value, this may create a bias with a devastating future results.

2.2.2.2 Normalize Input

Before we normalized our input, we will be creating 5 more columns by converting ['date'] into [['dayofweek', 'week', 'day', 'month', year']] to create a more diverse section in the same categorical inputs. Then, we will be converting and replacing our ['date'] data to a ['date'] version which is encoded in unix time stamp. Next, we will be normalizing our input [['date', 'year']] to create an input in a range of 0 to 1, in a floating point manner, this process is needed for machine learning algorithm especially ANN to better understand the data.

2.2.2.3 Encode Input

We will be encoding our data which is not a continuation type of input [['dayofweek', 'week', 'day', 'month']] into matrixes of 0 and 1 in an integer wise, to elaborate on why this is necessary, these types of inputs data are an approximate modular of (7, 52, 30 or 31, 12) respectively and by using these value directly in the data will cause the ANN model to think that the starting value and the ending value in the modular function is farther than in reality. As for the example the input data ['dayofweek'] in a range of 0 to 6, as we know day 0 and day 6 are only 1 day apart, however ANN might interpret this input data as being 6 days apart as $|0-6|$ is 6, which create a devastating bias in the real world meaning.

2.2.2.4 Split Dataset

At last before creating our model, we split the model into 4 part x_train, x_test, y_train, y_test to ensure that the model didn't see all the data and therefore accidentally overfitted our data, by splitting the model into train and test set, we can determine if the model is overfitted easier. We split the data into 70% on the train set and 30% on the test set, as recommend in the book Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow. We sorted our data by index and we are done in data preparation.

2.2.3 Neural Network

At last, we will be building ANN by using MLPRegressor from Sklearn module, we start by fitting our x_train and y_train inside the function and predict. Next, we visualize the data to see visually if we have overfitted or underfitted our model by the default Sklearn setting, and we will base it there on how we should further fitted out model. Once we get the clear picture, be using GridSearchCV to search through possibly great hyperparameters, after a few more adjustments toward adjusting hyperparameters, we will be testing it on our testing data, if it performs bad then we repeat adjusting the hyperparameters with GridSearchCV, else we are done.

3 Results

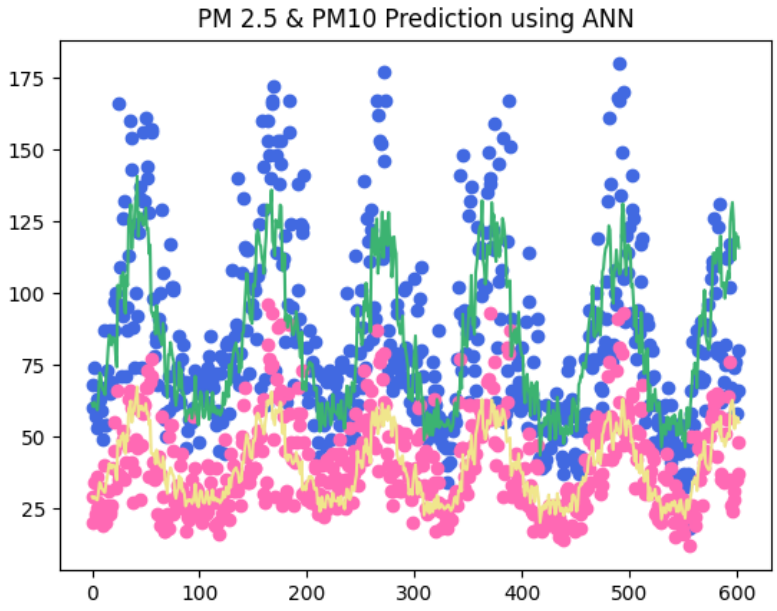

Figure 1: PM 2.5 & PM10 Prediction using ANN

This is the final outcome with this model as discuss in the Materials and Methods section:

PM 2.5 SQM Error : 46.76%

PM 10 SQM Error : 58.20%

Sum SQM Error : 104.97%.

4 Source Code

This Source Code is a source code shown on creating Figure 1.Note that this source code is not the same as in main source code but a similar replica with fewer codes. This is to ensure usage of space will be maintained at its minimum.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import this

df = pd.read_csv("bangkok-air-quality.csv")


"""
# # Data Preparation ##
"""
# Drop missing PM2.5 & PM10 Rows
# Note that We are Predicting PM25 and PM10, these two rows
    shouldn't be filled, else it will create a biased data.
df = df.replace('␣', np.nan)
df = df.dropna(subset=['␣pm25', '␣pm10', 'date'])

# String into Integer
df[['␣pm25', '␣pm10', '␣o3', '␣no2', '␣so2', '␣co']] = df[['␣
    pm25', '␣pm10', '␣o3', '␣no2', '␣so2', '␣co']].astype(
    float)
df[['␣pm25', '␣pm10', '␣o3', '␣no2', '␣so2', '␣co']] = df[['␣
    pm25', '␣pm10', '␣o3', '␣no2', '␣so2', '␣co']].astype(pd.
    Int32Dtype())

# Split Date into Day & Month & Year
df['date'] = pd.to_datetime(df['date'])

df['dayofweek'] = df['date'].dt.dayofweek
df['week'] = df['date'].dt.week
df['day'] = df['date'].dt.day
df['month'] = df['date'].dt.month
df['year'] = df['date'].dt.year

df['dayofweek'].head(3), df['week'].head(3), df['day'].head(3)
    , df['month'].head(3), df['year'].head(3), df['date'].head
    (3)

# Date Time to Unix Format
df['date'] = df['date'].astype(int)
```

```
# Normalized input
df['date'] = (df['date'] - df['date'].min()) / (df['date'].max
    () - df['date'].min())
df['year'] = (df['year'] - df['year'].min()) / (df['year'].max
    () - df['year'].min())

# Sort by index
df = df.sort_values(by='date')
df = df.reset_index(drop=True)

# Encoding
# Encoding Modular type
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder(sparse=False)

# encode
df_day_of_week = pd.DataFrame(encoder.fit_transform(df['
    dayofweek'].values.reshape(-1, 1)))
df_week = pd.DataFrame(encoder.fit_transform(df['week'].values
    .reshape(-1, 1)))
df_day = pd.DataFrame(encoder.fit_transform(df['day'].values.
    reshape(-1, 1)))
df_month = pd.DataFrame(encoder.fit_transform(df['month'].
    values.reshape(-1, 1)))

# index
df_day_of_week.index = df.index
df_week.index = df.index
df_day.index = df.index
df_month.index = df.index

# drop
df = df.drop(['dayofweek'], 1)
df = df.drop(['week'], 1)
df = df.drop(['day'], 1)
df = df.drop(['month'], 1)

# concat
df = pd.concat([df, df_day_of_week, df_week, df_day, df_month
    ], axis=1)

# drop extras
df = df.drop(['␣o3'], axis=1)
df = df.drop(['␣no2'], axis=1)
df = df.drop(['␣so2'], axis=1)
```

```
df = df.drop(['␣co'], axis=1)

df.shape , df.head()

# Split Train & Test Data
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(df.drop(['
    ␣pm25', '␣pm10'], axis=1), df[['␣pm25', '␣pm10']],
    test_size=0.3, random_state=42)

# Sort Data
x_train = x_train.sort_index()
x_test = x_test.sort_index()
y_train = y_train.sort_index()
y_test = y_test.sort_index()

"""
Neural Network
"""

from sklearn.neural_network import MLPRegressor
ANN = MLPRegressor(activation='relu', solver='adam',
    hidden_layer_sizes=(15, 15), max_iter=200,
    n_iter_no_change=2, tol=0.05, random_state=42, verbose=
    True)
ANN.fit(x_train.values.tolist(), y_train[['␣pm25', '␣pm10']].
    values.tolist())

prediction = ANN.predict(x_test)
prediction25 = prediction[:,0]
prediction10 = prediction[:,1]

result = pd.DataFrame({"Prediction25":prediction25, "Real␣
    Value25":y_test['␣pm25'].to_list(), "Prediction10":
    prediction10, "Real␣Value10":y_test['␣pm10'].tolist()})
result['Error␣Dec25'] = ((((result['Prediction25'] - result['
    Real␣Value25']) / result['Prediction25']).abs() + 1) ** 2)
     - 1
result['Error␣Dec10'] = ((((result['Prediction10'] - result['
    Real␣Value10']) / result['Prediction10']).abs() + 1) ** 2)
     - 1

averageerrorrate10 = result['Error␣Dec10'].mean()
averageerrorrate25 = result['Error␣Dec25'].mean()
```

```
print("PM␣2.5␣SQM␣Error␣Rate␣:␣%.4f%%"%(averageerrorrate25
    *100))
print("PM␣10␣SQM␣Error␣Rate␣:␣%.4f%%"%(averageerrorrate10*100)
    )


plt.scatter([i for i in range(len(prediction25))], result['
    Real␣Value25'].to_list(), color='b')
plt.plot([i for i in range(len(prediction25))], result['
    Prediction25'].to_list(), color='g')

plt.scatter([i for i in range(len(prediction10))], result['
    Real␣Value10'].to_list(), color='r')
plt.plot([i for i in range(len(prediction10))], result['
    Prediction10'].to_list(), color='y')
plt.title('PM␣2.5␣&␣PM10␣Prediction␣using␣ANN')
plt.show()

# # Save Model
# import joblib

# joblib.dump(ANN, 'ANN Model' + '.sav')
```