AIR QUALITY INDEX PREDICTION PROJECT REPORT

ENVIRONMENTAL PEACE AND SUSTAINABILITY

ASHISH ALICHEN 1127 7A
SWASTIK RAI 1172 7C
ANJANI PANDEY 1126 7B

Abstract

In recent years, many countries have experienced rapid urbanization. This has led to increased air pollution level and has created a major concern among the developing countries. Therefore, demand for predicting air quality has become a necessity so that we take precautionary measures to control the air pollution levels. This project is one specific contribution towards the issue of predicting air quality. The dataset we considered consists of major air pollutants as well as the meteorological data. We have used the standard formula followed by US EPA government for calculating the AQI using major pollutants and their concentrations. We also study the correlation and variation of one pollutant with respect to other pollutants.

We have applied machine learning algorithms like Multiple Linear Regression, Linear SVM, K-Nearest Neighbors and propose to apply Deep Learning models like CNN and LSTM to predict AQI (Air Quality Index. In LSTM, we will use past AQI and meteorological data to predict the next AQI value. The lesser the difference between the predicted and actual AQI value suggests a better model. We will further compare the above mentioned models to deduce the most efficient model. The results of the models were analyzed using evaluation metrics such as precision, recall, f1-score and accuracy. Among the models that we used, KNN outperformed the other two models by giving us a precision, recall, f1score, accuracy of 0.77, 0.82, 0.76, 81.77 % respectively.

Introduction

Due to rapid urbanization, many environmental hazards took place in 20th century, including rise in air pollution levels. Air pollution is the introduction of chemicals, particulate matter, or biological matter that cause harm or discomfort to living organisms or damage the natural environment or atmosphere. Air pollutants are tiny and light particles and thus they stay in the atmosphere for long duration and also easily bypass the filters of human nose and throat due to their small size. According to a recent survey, the presence of particulate matter has caused 4.2 million deaths. Major air pollutants like Sulphur Dioxide (SO2), Nitrogen Dioxide (NO2), Carbon Monoxide (CO), Particulate Matter (PM2.5, PM10) and Ozone(O3) have drastic effects on human health. Thus, predicting the air quality has become a major concern.

Particle size is critical in determining the particle deposition location in the human respiratory system. PM2.5, referring to particles with a diameter less than or equal to 2.5 μm, has been an increasing concern, as these particles can be deposited into the alveoli- the lung gas exchange region. The U.S. EPA revised the annual standard of PM2.5 by lowering the concentration to 12 μg/m3 to provide improved protection against health effects associated with long- and short-term exposure. Increased mortality and morbidity rates have been found in association with increased air pollutants. Thus, we considered PM2.5 as the label for classification.

Meteorological data is critical in determining the air pollutant consideration. The meteorological parameters considered by our model include: temperature, wind, relative humidity, dew point and pressure. Temperature affect air quality because of temperate inversion: the warm air above cooler air acts like a lid, suppressing vertical mixing and trapping the cooler air at the surface. As pollutants from vehicles, fireplaces, and industry are emitted into the air, the inversion traps these pollutants near the ground. Wind speed plays a big role in diluting pollutants. Generally, strong winds disperse pollutants, whereas light winds generally result in stagnant conditions allowing pollutants to build up over an area. Humidity could affect the diffusion of contaminant. Dew point indicates the amount of the moisture in the air. The higher the dew point the higher moisture content in the air at a given temperature. Dew point and the concentration of the air pollutants are inversely proportional. Pressure is also inversely proportional to the air quality.

The study of AQI started with the application of statistical models, but it was mostly been restricted to simply utilizing standard classification or regression models, which have neglected the nature of the problem itself or ignored the correlation between sub-models in different timeslots. The development of machine learning, helped to refine the model of a specific problem. This model predicts the levels of PM2.5 daily with the help of the meteorological data. For this we applied machine learning algorithms like SVM and deep learning techniques like ANN and LSTM. The model then compares and analyzes the results using the evaluation metrics and mean squared error. The model also helps us to study the correlation between various meteorological parameters and PM2.5.

LSTM is a specific recurrent neural network that doesn't only take the current input, but also what it has "perceived" previously in time, essentially using the output at time $t-1$ as an input to time t, along with the new input at time t. In our case, it is understood that the air quality of the current day does depend on the air quality of the previous day. But if we use right number of hidden layers and activation layers in the ANN we obtain slightly better results in same

period of time. So we can say that our ANN proved to be the best among the other models for pollution prediction.

Requirements

1. **Software requirements**
    a. Libraries used:
        i. Numpy
        ii. Pandas
        iii. Matplotlib
        iv. Sklearn
    b. Other Requirements
        i. Jupyter Notebook
        ii. Python

2. **Functional requirements**
    a. Calculation of AQI values using pollutant concentrations.
    b. Analyzing the relation between the air pollutants and meteorological data.
    c. Predicting the AQI values using the past data.

3. **Non-functional Requirements**
    a. Accuracy: It measures the ratio of correct predictions to the total number of data points evaluated. The Machine Learning models must have a significant amount of accuracy in prediction of apps.
    b. Recall: It is the fraction of relevant occurrences that have been retrieved over total amount of relevant occurrences. It is also known as the sensitivity of the model.
    c. Precision: It is the fraction of relevant occurrences among the retrieved occurrences. It is also referred as positive predictive value.

Code

```
#Importing libraries and loading dataset
import numpy as np
get_ipython().run_line_magic('matplotlib', 'inline')
import matplotlib.pyplot as plt
import pandas as pd
from datetime import datetime

def parse(x):
    return datetime.strptime(x, '%Y %m %d %H')

org_col_names=["No", "year","month", "day", "hour", "pm2.5",
"DEWP","TEMP", "PRES", "cbwd", "Iws", "Is", "Ir"]
col_names = ['pollution', 'dew', 'temp', 'pressure', 'w_dir',
'w_speed', 'snow', 'rain']

dataset = pd.read_csv('AirPollution.csv',index_col=0,
date_parser=parse,parse_dates=[['year', 'month', 'day',
'hour']])


# In[3]:


dataset.head()


# In[4]:


# Data cleaning
dataset.drop('No', axis=1, inplace=True)
dataset.columns = col_names
dataset['pollution'].fillna(0, inplace=True)
dataset = dataset[24:] # drop the first day
print(dataset.head(5))
dataset.to_csv('pollution.csv') # save new CSV


# In[5]:
```

```
# load dataset
df = pd.read_csv('pollution.csv', header=0, index_col=0)
df.describe()


# In[6]:


dataset_columns = df.columns.tolist()
dataset_columns


# In[7]:


#Box plot
#pd.options.display.mpl_style = False
df.boxplot()


# In[8]:


df


# In[9]:


#Analysing relationship between different features
# cor_cols = ['pollution', 'wnd_spd', 'rain', 'snow', 'temp']
cor_cols = dataset_columns
plt.matshow(df.corr())
plt.xticks(range(len(cor_cols)), cor_cols)
plt.yticks(range(len(cor_cols)), cor_cols)
plt.colorbar()
plt.show()


# In[10]:


from sklearn.preprocessing import LabelEncoder

# Encode non categorical values
```

```
values = df.values
encoder = LabelEncoder()
values[:,4] = encoder.fit_transform(values[:,4])
values = values.astype('float32')
values[:,4]


# In[11]:


#Normalising data
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0, 1))
scaled_dataset = scaler.fit_transform(values)


# In[12]:


#Creating a window for previous data
def to_supervised(train):
  window_size = 4
  X = []
  Y = []
  for i in range(window_size, len(train)):
    X.append(train[i-window_size:i,:])
    Y.append(train[i,0:1])

  return X,Y


# In[13]:


X, Y = to_supervised(scaled_dataset)
X = np.array(X)
Y = np.array(Y)
print('Y' ,Y.shape)
print('X' ,X.shape)


# In[14]:
```

```
#Splitting the dataset
n_train = 24*365
X_train, X_test = X[n_train:,] , X[:n_train,]
print('X_train' ,X_train.shape)
print('X_test' ,X_test.shape)

Y_train, Y_test = Y[n_train:,] , Y[:n_train,]
print('Y_train' ,Y_train.shape)
print('Y_test' ,Y_test.shape)


# In[15]:


#Importing LSTM model
from keras.models import Sequential
from keras.layers import Dense, Dropout,LSTM
model = Sequential()

model.add(LSTM(units = 50, return_sequences = True,
input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.2))

model.add(LSTM(units = 50, return_sequences = True))
model.add(Dropout(0.2))

model.add(LSTM(units = 50))
model.add(Dropout(0.2))

model.add(Dense(units = 1))

model.compile(optimizer = 'adam', loss =
'mean_squared_error')


# In[17]:


#Train the model
model.fit(X_train, Y_train, epochs = 10, batch_size = 32)


# In[18]:
```

```
#Taking predictions
from keras.models import load_model
model.save('AirPollutionMultivariate.h5')


# In[19]:


Y_pred = model.predict(X_test)

from sklearn.metrics import mean_squared_error
mse = mean_squared_error(Y_test, Y_pred)

rmse = np.sqrt(mse)
rmse


# In[20]:


#Y_predicted = scaler.inverse_transform(Y_pred)
print('y_predicted',Y_pred.shape)
#print('X_train' ,X_train.shape)
print('X_test' ,X_test.shape)
print('scaled Values shape', scaled_dataset.shape)


# In[21]:


#Scaling back to the original scale
d = scaled_dataset[:8760,:]
print('dummy',d.shape)
print('Y_pred',Y_pred.shape)
Y_predicted = np.concatenate((Y_pred,d[:8760,1:]), axis =1)
print('concat y_pred',Y_pred.shape)
Y_tested = np.concatenate((Y_test, d[:8760,1:]), axis = 1)
print('concat Y_test', Y_test.shape)


# In[22]:
```

```
Y_predicted = scaler.inverse_transform(Y_predicted)
Y_tested = scaler.inverse_transform(Y_tested)
Y_predicted = Y_predicted[:,0:1]
Y_tested = Y_tested[:,0:1]
print('Y_tested', Y_tested.shape)
print('Y_predicted', Y_predicted.shape)


# In[23]:


from sklearn.metrics import mean_squared_error
mse = mean_squared_error(Y_tested, Y_predicted)

rmse = np.sqrt(mse)
rmse


# In[27]:


#Plot the graph between actual vs predicted values
plt.figure(figsize=(10,6))
plt.plot(Y_predicted[:100,:], color= 'green',label =
'Predicted Pollution level')

plt.plot(Y_tested[:100,:] , color = 'red',label = 'Actual
Pollution level')
plt.title("Air Pollution Prediction (Multivariate)")
plt.xlabel("Days")
plt.ylabel("Pollution level")
plt.legend()
plt.show()
plt.savefig('graph.png')


# In[25]:


def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
```

```
mape = mean_absolute_percentage_error(Y_tested, Y_predicted)
print('MAPE', mape)


# In[26]:


from sklearn.metrics import mean_squared_error
mse = mean_squared_error(Y_tested, Y_predicted)
#print('Mean squared error', mse)
print('RMSE' , np.sqrt(mse))
print("Mean of Test data ", np.mean(Y_tested))


# In[ ]:
```

Result