MOISTURE MINDS REPORT

March 2023

1 Introduction

The two datasets we were given doesn't consists of all the parameters. The first dataset doesn't contain temperature,humidity,pressure columns.Likewise the second dataset doesn't contain soil temperature.We have to design a model which takes the dataset containing all parameters and predict the soil moisture.

2 Data pre-processing/ Data cleansing:

2.1 Code for pre-processing of the two datasets

- Look into your data
- Look at the proportion of missing data
- Check the data type of each column
- If you have columns of strings, check for trailing whitespaces
- Dealing with Missing Values (NaN Values)
- Extracting more information from your dataset to get more variables.Check the unique values of columns

The given datasets need to get merged such that each data point contains all parameter values.Initially after merging the two datasets, the temperature,humidity,pressure values for first dataset will be NaN and likewise we have soil temperature as NaN for second dataset. We have to pre-process the data such that the NaN values are replaced by reasonably well-fitting data so that our model can be trained from good data and predict more accurately. Now, we are filling up those missing data by computing the mean of the parameters over one month for more precise values.And dropping the rows which have whitespaces in them.We will be also sorting the dataframe in the pre-processing.

```
import pandas as pd
df=pd.read_csv("https://raw.githubusercontent.com/chidaksh/CosmosocClub/master/Parsec2023/us
df2=pd.read_csv("https://raw.githubusercontent.com/chidaksh/CosmosocClub/master/Parsec2023/u

#combining the given two data frames
df3 = pd.concat([df, df2], ignore_index=True)

df3['ttime'] = pd.to_datetime(df3['ttime'])

df3['month'] = df3['ttime'].dt.month
df3=df3.drop('ttime',axis=1)

import numpy as np
st_means = df3.groupby('month')['st'].mean()
for i in range(len(df3)):
    if(np.isnan(df3['st'][i])):
        df3['st'][i]=st_means[df3['month'][i]]

import numpy as np
temp_means = df3.groupby('month')['temp'].mean()
for i in range(len(df3)):
    if(np.isnan(df3['temp'][i])):
        df3['temp'][i]=temp_means[df3['month'][i]]

import numpy as np
humd_means = df3.groupby('month')['humd'].mean()
for i in range(len(df3)):
    if(np.isnan(df3['humd'][i])):
        df3['humd'][i]=humd_means[df3['month'][i]]

import numpy as np
pres_means = df3.groupby('month')['pres'].mean()
for i in range(len(df3)):
    if(np.isnan(df3['pres'][i])):
        df3['pres'][i]=pres_means[df3['month'][i]]

df3.isnull().sum()

months=[7,8,9,10,11,12,1,2,3]
indices=[0,1,2,3,4,5,6,7,8,9]
for i in range(len(df3)):
    if(df3['month'][i]==7):
        df3['month'][i]=0
    if(df3['month'][i]==8):
        df3['month'][i]=1
    if(df3['month'][i]==9):
        df3['month'][i]=2
    if(df3['month'][i]==10):
        df3['month'][i]=3
    if(df3['month'][i]==11):
        df3['month'][i]=4
    if(df3['month'][i]==12):
        df3['month'][i]=5
    if(df3['month'][i]==1):
        df3['month'][i]=6
    if(df3['month'][i]==2):
        df3['month'][i]=7
    if(df3['month'][i]==3):
        df3['month'][i]=8

df3=df3.sort_values(by=['month'])
df3 = df3.reset_index(drop=True)
df3

df3=df3.drop('month',axis=1)
df3.head()

df3.info()
x=df3.drop('sm',axis=1)
y=pd.DataFrame(df3['sm'])
```

2.2 Decision Tree Regressor Model:

Decision Tree is one of the most commonly used, practical approaches for supervised learning. It can be used to solve both Regression and Classification tasks with the latter being put more into practical application.

It is a tree-structured classifier with three types of nodes. The Root Node is the initial node which represents the entire sample and may get split further into further nodes. The Interior Nodes represent the features of a data set and the branches represent the decision rules. Finally, the Leaf Nodes represent the outcome. This algorithm is very useful for solving decision-related problems.

With a particular data point, it is run completely through the entirely tree by answering True/False questions till it reaches the leaf node. The final prediction is the average of the value of the dependent variable in that particular leaf node. Through multiple iterations, the Tree is able to predict a proper value for the data point.

Decision trees have an advantage that it is easy to understand, lesser data cleaning is required, non-linearity does not affect the model's performance and the number of hyper-parameters to be tuned is almost null. However, it may have an over-fitting problem, which can be resolved using the Random Forest algorithm.

Here we are predicting the soil moisture using Decision tree regression model, based on the dataset information or features we were given.

If we were given the dataset regarding the soil moisture dates that we have to predict then we can predict it using the model.Since,the parameter that has to be predicted is soil moisture it woyuld be high in months july-october and it gradually decreases in winter (i.e., october-february) and will be low in summer(march-june).

Code for Decision Tree regressor

```
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
x_scaled=scale.fit_transform(x)
x_scaled=pd.DataFrame(x_scaled)
x_scaled.columns=x.columns
x_scaled.head()

from sklearn.tree import DecisionTreeRegressor
dtr=DecisionTreeRegressor()
dtr.fit(x,y)
y_pred=dtr.predict(x)

import matplotlib.pyplot as plt
plt.figure()
plt.plot(y,'.')
#plt.plot(y_pred,color='green')
plt.show()
```