Report

Project : Soil Moisture Prediction

Team : Bubbles

Objective

The objective of this project is to predict soil moisture using machine learning algorithms. The dataset used in this project is collected from two different users and then merged based on the timestamp column. The dataset is then cleaned and pre-processed to be fed into the machine learning model. The model used in this project is Multiple Linear Regression (MLR) to predict soil moisture.

Requirements

Files

- `user1_data.csv`: contains data from user 1.
- `user2_data.csv`: contains data from user 2.

Dependencies

This code requires the following dependencies:

- numpy
- pandas
- sklearn

Data Loading

The first step is to load the necessary libraries and then read in the two CSV files containing the data. We use Pandas library to read the data from CSV files into dataframes:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

df1 = pd.read_csv('user2_data.csv',header=[0])
df2 = pd.read_csv('user1_data.csv',header=[0])
```

Data Operations

The timestamp columns in the two dataframes are converted to datetime format using the pd.to_datetime() method. Then, the old timestamp columns are dropped, and the new datetime columns are set as the index of the dataframes. The data is then resampled to a common time interval of 5 minutes using the resample() method. Finally, the two dataframes are merged on the timestamp column to create a single, clean dataframe:

```
df1['timestamp'] = pd.to_datetime(df1['ttime'])
df2['timestamp'] = pd.to_datetime(df2['ttime'])
```

```
df1.drop(['ttime'],axis=1,inplace=True)
df2.drop(['ttime'],axis=1,inplace=True)

df1.set_index('timestamp', inplace=True)
df2.set_index('timestamp', inplace=True)

df1_resampled = df1.resample('5T').mean()
df2_resampled = df2.resample('5T').mean()
merged_df = pd.merge(df1_resampled, df2_resampled, on='timestamp')
merged_df.to_csv("new_data.csv")
```

Data Cleaning

The cleaned data is then read back into a new dataframe using Pandas. We replace the timestamp column with a new_time column containing Unix timestamps. We then drop the timestamp column and any rows containing NaN values. Finally, we drop the sm_x column as this is the dependent variable we will be trying to predict:

```
df = pd.read_csv('new_data.csv',header=[0])

df['new_time'] = df['timestamp'].apply(lambda x: pd.Timestamp(x).timestamp())
df.drop(['timestamp'],axis=1,inplace=True)
```

```
df= df.replace(-99,np.nan)
df=df.dropna()
df.columns
df.head()
f = df.drop(['sm_x'], axis = 1)
x = f
y=df['sm_x']
```

Data Preprocessing

Next, we split the data into training and testing sets using the train_test_split() method. We then scale the independent variables using the StandardScaler() method:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y,
test_size = 0.25, random_state = 0)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Model Building and Evaluation

We then build a multiple linear regression model using the LinearRegression() method from the scikit-learn library. We fit the model to the training data and use it to predict the dependent variable for the test data. Finally, we evaluate the model using the RMSE metric:

```
from sklearn.linear_model import LinearRegression
classifier = LinearRegression()
```