RPL_weather

July 18, 2023

1 import library and Splitting data for training the model

```
[ ]: # Importing pandas library
     import pandas as pd

     # Reading weather forecast data from csv file
     weather_data = pd.read_csv('train.csv')
     # weather_data

     # Converting date column to the appropriate format
     weather_data['date'] = pd.to_datetime(weather_data['date'])

     # Extracting month from the date
     weather_data['month'] = weather_data['date'].dt.month

     # Separating rows related to months from November to December
     weather_data_Nov_Dec = weather_data.loc[(weather_data['month'] >= 11) &␣
      ↪(weather_data['month'] <= 12)]

     # Displaying the separated data
     print(weather_data_Nov_Dec)
```

```
           date  precipitation  temp_max  temp_min  wind weather  month
305  2012-11-01            9.7      15.0      10.6   3.0    rain     11
306  2012-11-02            5.6      15.0      10.6   1.0    rain     11
307  2012-11-03            0.5      15.6      11.1   3.6    rain     11
308  2012-11-04            8.1      17.8      12.8   3.8    rain     11
309  2012-11-05            0.8      15.0       7.8   4.0    rain     11
…           …             …       …         …     …      …       …
1091 2014-12-27            3.3       9.4       4.4   4.9    rain     12
1092 2014-12-28            4.1       6.7       2.8   1.8    rain     12
1093 2014-12-29            0.0       6.1       0.6   4.3     fog     12
1094 2014-12-30            0.0       3.3      -2.1   3.6     sun     12
1095 2014-12-31            0.0       3.3      -2.7   3.0     sun     12

[183 rows x 7 columns]
```

2 Check have null data

"Cleaning the data is the first step. Since the number of null values for any column is 0, we do not have any null values and therefore do not need to clean the data."

```
[ ]: weather_data_Nov_Dec.isnull().sum()
```

```
[ ]: date             0
     precipitation    0
     temp_max         0
     temp_min         0
     wind             0
     weather          0
     month            0
     dtype: int64
```

3 Upload the final_test.csv to add weather column

```
[ ]: df_final = pd.read_csv('final_test.csv')
     df_final.isnull().sum()
     df_final['weather'] = '' # create the weather column of final_test.csv
     df_final.head()
```

```
[ ]:          date  precipitation  temp_max  temp_min  wind weather
     0  12/11/2015            0.3       9.4       4.4   2.8
     1   12/9/2015           13.5      12.2       7.8   6.3
     2  11/26/2015            0.0       9.4      -1.0   4.3
     3  12/31/2015            0.0       5.6      -2.1   3.5
     4   12/1/2015           12.2      10.0       3.9   3.5
```

4 Preprocessing data (encoding data)

We cannot use string values for training, so we need to preprocess and convert them to numerical values (encode the data), which involves converting the categorical variables to numerical values.

```
[ ]: from sklearn import preprocessing
     le = preprocessing.LabelEncoder()
     weather_data_Nov_Dec['date'] = le.fit_transform(weather_data_Nov_Dec['date'])
     weather_data_Nov_Dec['precipitation'] = le.
      ↪fit_transform(weather_data_Nov_Dec['precipitation'])
     weather_data_Nov_Dec['temp_max'] = le.
      ↪fit_transform(weather_data_Nov_Dec['temp_max'])
     weather_data_Nov_Dec['temp_min'] = le.
      ↪fit_transform(weather_data_Nov_Dec['temp_min'])
     weather_data_Nov_Dec['wind'] = le.fit_transform(weather_data_Nov_Dec['wind'])
     weather_data_Nov_Dec['weather'] = le.
      ↪fit_transform(weather_data_Nov_Dec['weather'])
```

```
C:\Users\abasi\AppData\Local\Temp\ipykernel_11588\3828091490.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  weather_data_Nov_Dec['date'] = le.fit_transform(weather_data_Nov_Dec['date'])
C:\Users\abasi\AppData\Local\Temp\ipykernel_11588\3828091490.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  weather_data_Nov_Dec['precipitation'] =
le.fit_transform(weather_data_Nov_Dec['precipitation'])
C:\Users\abasi\AppData\Local\Temp\ipykernel_11588\3828091490.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  weather_data_Nov_Dec['temp_max'] =
le.fit_transform(weather_data_Nov_Dec['temp_max'])
C:\Users\abasi\AppData\Local\Temp\ipykernel_11588\3828091490.py:6:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  weather_data_Nov_Dec['temp_min'] =
le.fit_transform(weather_data_Nov_Dec['temp_min'])
C:\Users\abasi\AppData\Local\Temp\ipykernel_11588\3828091490.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  weather_data_Nov_Dec['wind'] = le.fit_transform(weather_data_Nov_Dec['wind'])
C:\Users\abasi\AppData\Local\Temp\ipykernel_11588\3828091490.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  weather_data_Nov_Dec['weather'] =
le.fit_transform(weather_data_Nov_Dec['weather'])
```

5 which number belongs to which weather

changing the data type for better performance and memory usage optimization.

[ ]:
```
# 'drizzle' = 0 , 'rain' = 2 , 'sun' = 4 , 'snow' = 3 , 'fog' = 1
unique_weather = weather_data_Nov_Dec['weather'].unique().astype('int16')
unique_weather
```

[ ]:
```
array([2, 4, 0, 1, 3], dtype=int16)
```

6 We should define the label and remove 'date' from the features.

Here weather is our label and we want predict that with classification

[ ]:
```
cols = [col for col in df_final.columns if col not in ['weather','date']]

data = weather_data_Nov_Dec[cols]
target = weather_data_Nov_Dec['weather']
```

7 Training

[ ]:
```
from sklearn.model_selection import train_test_split
data_train, data_test, target_train, target_test = train_test_split(data,␣
 ↪target, train_size=0.999, test_size=0.001)
```

8 Dictionary values

This dictionary is used to set the values of the encoding to strings.

[ ]:
```
values = {
     0 :'drizzle',
     2 :'rain' ,
     4 :'sun'  ,
     3 :'snow'  ,
     1 :'fog'
}

df_final
```

[ ]:
```
         date  precipitation  temp_max  temp_min  wind weather
0  12/11/2015            0.3       9.4       4.4   2.8
```

```
1    12/9/2015          13.5      12.2       7.8   6.3
2   11/26/2015           0.0       9.4      -1.0   4.3
3   12/31/2015           0.0       5.6      -2.1   3.5
4    12/1/2015          12.2      10.0       3.9   3.5
..         …           …       …        …    …     …
57  11/16/2015           2.0       8.9       1.7   4.0
58  12/22/2015           4.6       7.8       2.8   5.0
59  12/21/2015          27.4       5.6       2.8   4.3
60  11/22/2015           0.0      10.0       1.7   3.1
61  12/27/2015           8.6       4.4       1.7   2.9

[62 rows x 6 columns]
```

9 Naive Bayes Model

[ ]:
```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
ghb_model = GaussianNB()
ghb_model.fit(data_train, target_train)
pred = ghb_model.predict(data_test)
print('Navy bayes accurency: ', accuracy_score(target_test, pred,␣
 ↪normalize=True)) # evaluation of the model of train

# for predict the weather of finaal file & save RPL_weather.csv

predicted_weather = ghb_model.predict(df_final[cols]) # for fill the weather␣
 ↪column in final_test.csv
df_final['weather'] = predicted_weather

df_final['weather'] = df_final['weather'].map(values) # Set the values of␣
 ↪weather based on the above dictionary.

df_final.to_csv("RPL_weather.csv", index=False) # for create and save file
df_final

# save the weather column
df_weather = df_final['weather']
df_weather.to_csv("RPL_weather_column.csv", index=False)
df_weather
```

```
Navy bayes accurency:  1.0
```

[ ]:
```
0     rain
1     rain
2      sun
3      sun
4     rain
```

```
      …
57    rain
58    rain
59    rain
60     sun
61    rain
Name: weather, Length: 62, dtype: object
```