Data_cleaning

September 7, 2021

[3]:
```
%run -i process.py
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

0.1 Load files

[4]:
```
## Load the NOAA ##

filename = r"NOAA_SAN_MARCOS.csv"

df_SM_NOAA =pd.read_csv(filename)
df_SM_NOAA.index = pd.DatetimeIndex(df_SM_NOAA['DATE'])
df_SM_NOAA = df_SM_NOAA.drop(columns =[ 'STATION','NAME','LATITUDE'
                                       ,'LONGITUDE','ELEVATION','DATE'])
df_Dateindex = df_SM_NOAA.index
###------Make the Date Range equal---------------------###
df_SM_NOAA = (df_SM_NOAA[(df_SM_NOAA.index >='1960-01-01')
                         &(df_SM_NOAA.index <= '2020-12-31')])

print('San Marcos')
## Statistics and split the dataframe >= Prism & <= Prism
df_range = pd.date_range(start=df_SM_NOAA.index[0], end=df_SM_NOAA.index[-1])

###---From the data range obtain the missing values-------------#df_PRISM###
missing_values = df_range.difference(df_SM_NOAA.index)
#missing_values

df_range = pd.DataFrame(df_range, columns=['DATE'], index = None)
df_range.columns = ['DATE']
df_range.index = df_range['DATE']

df_SM_NOAA = pd.concat([df_range, df_SM_NOAA], axis=1, join='outer')
df_SM_NOAA = df_SM_NOAA.drop(columns='DATE')

status(df_SM_NOAA)
```

```
##Load Prism

filename = r"PRISM_.csv"
base_dir = os.getcwd()
df_PRISM =pd.read_csv(os.path.join(base_dir,filename))
#indexer.index = indexer['DATE']
df_PRISM = df_PRISM.rename(columns = {'Date':'DATE'})
df_PRISM.index = pd.DatetimeIndex(df_PRISM.DATE)
df_PRISM=df_PRISM.drop(columns='DATE')
df_PRISM =df_PRISM.set_axis(['P_PRCP', 'P_TMIN', 'P_TMAX'], axis='columns')
df_PRISM = df_PRISM[df_PRISM.index < '2020-12-31']

print('San Marcos_PRISM')

status(df_PRISM)
```

```
San Marcos
+----------+--------------+---------------+------------+
| Features | Observations | No of missing | % Missing  |
+----------+--------------+---------------+------------+
|   PRCP   |    22281     |     1087      |    4.9%    |
|   TMAX   |    22281     |      858      |    3.9%    |
|   TMIN   |    22281     |      972      |    4.4%    |
+----------+--------------+---------------+------------+
San Marcos_PRISM
+----------+--------------+---------------+------------+
| Features | Observations | No of missing | % Missing  |
+----------+--------------+---------------+------------+
|  P_PRCP  |    14609     |       0       |    0.0%    |
|  P_TMIN  |    14609     |       0       |    0.0%    |
|  P_TMAX  |    14609     |       0       |    0.0%    |
+----------+--------------+---------------+------------+
```

0.2 Before Prism data

[5]:
```
## Get the individual dataframe before the prism
df_BF = df_SM_NOAA[df_SM_NOAA.index < df_PRISM.index[0]]
Tmax = df_BF.drop(columns = ['PRCP','TMIN'])
Tmin = df_BF.drop(columns = ['PRCP','TMAX'])
PRCP = df_BF.drop(columns = ['TMIN','TMAX'])
status(df_BF)
```

```
+----------+--------------+---------------+------------+
| Features | Observations | No of missing | % Missing  |
+----------+--------------+---------------+------------+
|   PRCP   |     7671     |      185      |    2.4%    |
|   TMAX   |     7671     |      234      |    3.1%    |
|   TMIN   |     7671     |      352      |    4.6%    |
```

```
+----------+-------------+--------------+------------+
```

0.3 Month with missing sequence

[6]:
```
#Include the Month df_PRISM
df_SM_NOAA.assign(Month= df_SM_NOAA.index.month)

## Test a model with the extra features added
"""Obtain the largest missing sequence we have"""
from itertools import groupby
df =df_BF
df = df.fillna(-999)
"""Create the table of the missing range"""
table = [[v.index[0],
          v.index[-1],len(v)]for k, v in df[df['PRCP'] == -999]
         .groupby((df['PRCP'] != -999).cumsum())]



df_missing = pd.DataFrame(table, columns=['start_Date','End_Date','Frequency'])
df_missing.sort_values(by = ['Frequency'] , ascending= False).head(20)
```

[6]:

	start_Date	End_Date	Frequency
3	1967-01-01	1967-02-28	59
1	1965-04-01	1965-04-30	30
2	1966-11-01	1966-11-30	30
5	1974-06-01	1974-06-30	30
0	1961-09-01	1961-09-17	17
4	1974-01-01	1974-01-17	17
6	1977-01-01	1977-01-02	2

0.4 Test the KNNImputer and the linear Interpolation

[7]:
```
from sklearn.impute import KNNImputer
df_BF=df_BF.assign(Month=df_BF.index.month)
status(df_BF)
```

```
+----------+-------------+---------------+------------+
| Features | Observations | No of missing | % Missing  |
+----------+-------------+---------------+------------+
|   PRCP   |     7671     |      185      |    2.4%    |
|   TMAX   |     7671     |      234      |    3.1%    |
|   TMIN   |     7671     |      352      |    4.6%    |
|  Month   |     7671     |       0       |    0.0%    |
+----------+-------------+---------------+------------+
```

[8]:
```
Imputer = KNNImputer(weights='distance',n_neighbors = 10)
Knn = Imputer.fit_transform(df_BF)
Knn = pd.DataFrame(Knn, columns = ['PRCP','TMAX','TMIN', 'Month'], index =␣
↪df_BF.index)
```

0.5 Linear Interpolate

[9]:
```
df_BF= df_BF.interpolate(method = 'linear', limit_direction= 'both')
```

0.6 KNN R-squared score

[10]:
```
print(r2_score(Knn['TMAX'],df_BF['TMAX']))
print(r2_score(Knn['TMIN'],df_BF['TMIN']))
print(r2_score(Knn['PRCP'],df_BF['PRCP']))
```

```
0.985935239094394
0.9815344567361339
0.9981934707508145
```

[11]:
```
status(df_BF)
```

```
+----------+--------------+---------------+------------+
| Features | Observations | No of missing | % Missing  |
+----------+--------------+---------------+------------+
|   PRCP   |     7671     |       0       |    0.0%    |
|   TMAX   |     7671     |       0       |    0.0%    |
|   TMIN   |     7671     |       0       |    0.0%    |
|  Month   |     7671     |       0       |    0.0%    |
+----------+--------------+---------------+------------+
```

1 Seperate each feature from 1981

Drop the na from prism date range

[12]:
```
Tmax = df_SM_NOAA[df_SM_NOAA.index.year >= 1981].drop(columns= ['TMIN','PRCP']).
↪dropna()
Tmin = df_SM_NOAA[df_SM_NOAA.index.year >= 1981].drop(columns= ['TMAX','PRCP']).
↪dropna()
Prcp = df_SM_NOAA[df_SM_NOAA.index.year >= 1981].drop(columns= ['TMAX','TMIN']).
↪dropna()
```

[13]:
```
status(df_SM_NOAA[df_SM_NOAA.index.year >= 1981])
```

```
+----------+--------------+---------------+------------+
| Features | Observations | No of missing | % Missing  |
+----------+--------------+---------------+------------+
|   PRCP   |    14610     |      902      |    6.2%    |
```

```
|   TMAX   |    14610     |      624      |    4.3%    |
|   TMIN   |    14610     |      620      |    4.2%    |
+----------+--------------+---------------+------------+
```

1.1 Check the predictive accuracy of the prism data on the NOAA

1.2 Tmax

[14]:
```
df_Tmax, Tmax_pipeline = pipeline(Tmax, df_PRISM['P_TMAX'])
```

```
+-------------------------+
|    Training set Score   |
+--------------+----------+
|  Algorithm   | R-square |
+--------------+----------+
| RandomForest |  84.12%  |
|   XGBoost    |  84.10%  |
|  ExtraTree   |  84.15%  |
+--------------+----------+
+-------------------------+
|      Test set Score     |
+--------------+----------+
|  Algorithm   | R-square |
+--------------+----------+
| RandomForest |  80.40%  |
|   XGBoost    |  80.46%  |
|  ExtraTree   |  80.43%  |
+--------------+----------+
```

1.3 Tmin

[15]:
```
df_Tmin, Tmin_pipeline = pipeline(Tmin, df_PRISM['P_TMIN'])
```

```
+-------------------------+
|    Training set Score   |
+--------------+----------+
|  Algorithm   | R-square |
+--------------+----------+
| RandomForest |  92.34%  |
|   XGBoost    |  92.35%  |
|  ExtraTree   |  92.35%  |
+--------------+----------+
+-------------------------+
|      Test set Score     |
+--------------+----------+
|  Algorithm   | R-square |
+--------------+----------+
| RandomForest |  92.26%  |
|   XGBoost    |  92.25%  |
```

```
|  ExtraTree    |  92.21%  |
+--------------+----------+
```

1.4 PRCP

All three features give R-square score > 50

[16]:
```
df_P = join_first_df(Prcp,df_PRISM)
```

[17]:
```
Feature_Names = df_PRISM.columns
per_of_len = int(len(df_P)*.9)


df_Train, df_Test =df_P.iloc[:per_of_len,:],df_P.iloc[per_of_len:,:]


X_train = df_Train[Feature_Names].to_numpy()
X_test = df_Test[Feature_Names].to_numpy()
y_train = df_Train['PRCP'].to_numpy()
y_test = df_Test['PRCP'].to_numpy()
## Build the pipelines
```

[18]:
```
from sklearn.pipeline import Pipeline
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV


p_grid = dict(n_estimators = [int(i) for i in np.linspace(100,2000,num=20)],
              max_depth = [int(i) for i in np.linspace(6,12,num=7)])

model = RandomizedSearchCV(estimator = ExtraTreesRegressor(),
                           param_distributions= p_grid,
                           scoring = r2_score,
                           cv = 3, verbose=1, n_jobs=-1)


model = model.fit(X_train,y_train)




print_results(model,X_train,y_train,X_test,y_test)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
############ Model Train Accuracy ############
RMSE: 5.690 mm.
MAE: 1.382 mm.
R-squared: 0.715
```

```
MAPE: inf
########################################
############ Model Test Accuracy  ############
RMSE: 6.122 mm.
MAE: 1.794 mm.
R-squared: 0.685
MAPE: inf
########################################
```

[17]:
```
model = ExtraTreesRegressor(**model.best_params_).fit(X_train,y_train)
```

1.5 Use models to predict missing dates > 1981

individual features

[18]:
```
TMAX = df_SM_NOAA[df_SM_NOAA.index.year >= 1981].drop(columns= ['TMIN','PRCP'])
TMIN = df_SM_NOAA[df_SM_NOAA.index.year >= 1981].drop(columns= ['TMAX','PRCP'])
PRCP = df_SM_NOAA[df_SM_NOAA.index.year >= 1981].drop(columns= ['TMAX','TMIN'])
```

[19]:
```
Missing_TMIN = pd.DataFrame(df_PRISM.loc[TMIN[TMIN["TMIN"].isna()].
 ↪index]["P_TMIN"])
Missing_TMAX = pd.DataFrame(df_PRISM.loc[TMAX[TMAX["TMAX"].isna()].
 ↪index]["P_TMAX"])
Missing_PRCP = pd.DataFrame(df_PRISM.loc[TMAX[PRCP["PRCP"].isna()].index])
```

ExtraTree model is useds to predict the mising values of precipitation with the use of precipitation from prism, older analysis

[20]:
```
FIll = pd.DataFrame({'TMIN':Tmin_pipeline[2].predict(Missing_TMIN)}
                    , index = Missing_TMIN.index)
FIll2 = pd.DataFrame({'TMAX':Tmax_pipeline[2].predict(Missing_TMAX)}
                     , index = Missing_TMAX.index)



FIll3 = pd.DataFrame({'PRCP':model.predict(Missing_PRCP)}
                     , index = Missing_PRCP.index)
FIll3.loc[(FIll3['PRCP'] < 1)] = pd.DataFrame(df_PRISM['P_PRCP']
                                              .loc[FIll3[FIll3['PRCP'] < 1].
 ↪index])
```

[21]:
```
TMAX.loc[FIll2.index] = FIll2
TMIN.loc[FIll.index] = FIll
PRCP.loc[FIll3.index] = FIll3
```

[22]:
```
df_CLIM_1981 = pd.concat([TMAX,TMIN,PRCP],axis = 1, join='outer')
df_CLIM_1981= df_CLIM_1981.assign(Month = df_CLIM_1981.index.month)
```

[23]:
```
## append to the data < 1981
```

[24]:
```
status(df_BF)
```

```
+----------+--------------+---------------+------------+
| Features | Observations | No of missing | % Missing  |
+----------+--------------+---------------+------------+
|   PRCP   |     7427     |       0       |    0.0%    |
|   TMAX   |     7427     |       0       |    0.0%    |
|   TMIN   |     7427     |       0       |    0.0%    |
|  Month   |     7427     |       0       |    0.0%    |
+----------+--------------+---------------+------------+
```

[25]:
```
df_SM = df_BF.append(df_CLIM_1981)


df_SM.to_csv('df_SM.csv')
```

[26]:
```
import pandas as pd
import numpy as np
from IPython.display import display, HTML

CSS = """
.output {
    flex-direction: row;
}
"""

HTML('<style>{}</style>'.format(CSS))
```

[26]:
```
<IPython.core.display.HTML object>
```

Load the San marcos springs data

[27]:
```
skip_rows = 32
#base_dir  = r"C:\Users\hbasagaoglu\Documents\Springs_EAA\SanMarcos_Springs"
filename  = r"Daily_SpringDischarge_SanMarcos.csv"

SanMarcos_Spring_Flow_or = pd.read_csv('Daily_SpringDischarge_SanMarcos.csv',
                                       skiprows = 30)

SanMarcos_Spring_Flow = SanMarcos_Spring_Flow_or[['20d',
                                                  '14n']].copy()
SanMarcos_Spring_Flow = SanMarcos_Spring_Flow.rename(
    columns = {'20d' : 'Date', '14n' : 'SanMarcos_SF(cfs)'})

SanMarcos_Spring_Flow.index=  pd.to_datetime(SanMarcos_Spring_Flow['Date'])
```

```
SanMarcos_Spring_Flow['SanMarcos_SF(m3)'] =␣
↪round(SanMarcos_Spring_Flow['SanMarcos_SF(cfs)']
                                                .astype(float)*0.028316847, 3)


SanMarcos_Spring_Flow.drop(columns=['Date'])
```

[27]:

Date	SanMarcos_SF(cfs)	SanMarcos_SF(m3)
1956-05-26	65.0	1.841
1956-05-27	69.0	1.954
1956-05-28	69.0	1.954
1956-05-29	68.0	1.926
1956-05-30	65.0	1.841
…	…	…
2021-03-04	112.0	3.171
2021-03-05	112.0	3.171
2021-03-06	112.0	3.171
2021-03-07	113.0	3.200
2021-03-08	112.0	3.171

[23663 rows x 2 columns]

[28]:
```
Sanmarcos_sf = SanMarcos_Spring_Flow.loc[(SanMarcos_Spring_Flow.index␣
↪>='1960-09-01')
                                         &(SanMarcos_Spring_Flow.index␣
↪<='2020-12-31')]



Sanmarcos_sf= Sanmarcos_sf.interpolate(method = 'linear', limit_direction=␣
↪'both')

status(Sanmarcos_sf)
```

```
+------------------+--------------+---------------+------------+
|     Features     | Observations | No of missing | % Missing  |
+------------------+--------------+---------------+------------+
|       Date       |    22037     |       0       |    0.0%    |
| SanMarcos_SF(cfs) |    22037     |       0       |    0.0%    |
| SanMarcos_SF(m3)  |    22037     |       0       |    0.0%    |
+------------------+--------------+---------------+------------+
```

[29]:
```
len(df_SM)
```

[29]: 22037

create the dataframe

[30]:
```
df_AI = pd.concat([df_SM,Sanmarcos_sf.
↪drop(columns=['Date','SanMarcos_SF(cfs)'])]
          ,axis = 1,join='outer')

df_AI.index.rename('DATE')
```

[30]:
```
DatetimeIndex(['1960-09-01', '1960-09-02', '1960-09-03', '1960-09-04',
               '1960-09-05', '1960-09-06', '1960-09-07', '1960-09-08',
               '1960-09-09', '1960-09-10',
               …
               '2020-12-22', '2020-12-23', '2020-12-24', '2020-12-25',
               '2020-12-26', '2020-12-27', '2020-12-28', '2020-12-29',
               '2020-12-30', '2020-12-31'],
              dtype='datetime64[ns]', name='DATE', length=22037, freq='D')
```

[31]:
```
df_AI.to_csv('df_AI.csv')
```

Prepare XAI framework