Milestone3-Task3

April 16, 2022

1 Task 3

2 Imports

[1]:
```
import numpy as np
import pandas as pd
from joblib import dump, load
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split, cross_validate
import matplotlib.pyplot as plt
plt.style.use('ggplot')
plt.rcParams.update({'font.size': 16, 'axes.labelweight': 'bold', 'figure.
 ↪figsize': (8,6)})
## add any other additional packages that you need. You are free to use any␣
 ↪packages for vizualization.
import altair as alt
alt.data_transformers.disable_max_rows()
```

[1]:
```
DataTransformerRegistry.enable('default')
```

2.1 Part 1:

Recall as a final goal of this project. We want to build and deploy ensemble machine learning models in the cloud, where features are outputs of different climate models and the target is the actual rainfall observation. In this milestone, you'll actually build these ensemble machine learning models in the cloud.

Your tasks:

1. Read the data CSV from your s3 bucket.
2. Drop rows with nans.
3. Split the data into train (80%) and test (20%) portions with `random_state=123`.
4. Carry out EDA of your choice on the train split.
5. Train ensemble machine learning model using `RandomForestRegressor` and evaluate with metric of your choice (e.g., `RMSE`) by considering `Observed` as the target column.
6. Discuss your results. Are you getting better results with ensemble models compared to the individual climate models?

Recall that individual columns in the data are predictions of different climate models.

[2]:
```
## Depending on the permissions that you provided to your bucket you might need␣
↪to provide your aws credentials
## to read from the bucket, if so provide with your credentials and pass as␣
↪storage_options=aws_credentials
aws_credentials = {"key" : "ASIARS6J4JEAH2ZOBKCD",
                   "secret": "vhJG5S0k783tWBCqOZlkY8NTwaXZl1jMtfq8gGjT",
                   "token": "FwoGZXIvYXdzELj//////////
↪wEaDLBw9pd5Kse5Pi7LiyLCAasr9ifix3bKDd6nf5hHHCxKTYtX7shxFfRM73uJApDrJmPlhXN1bQ1swlqSJjGrGoly
↪waunsP2WlwU2ygC3iiBz2/
↪y1d3UhV+67jqnXdkUQIxVy8nN0DOv7u6IErx8VlqzbTynpMODyXeuBTJzLlVWhQtPQ8chuQIIR9aFi+hpKg28x+0mbV
                  }

#df = pd.read_csv("s3://mds-s3-group18/output/ml_data_SYD.csv",␣
↪storage_options=aws_credentials, index_col=0, parse_dates=True)
df = pd.read_csv("ml_data_SYD.csv")
```

[3]:
```
df
```

[3]:
```
             time  ACCESS-CM2_rainfall  ACCESS-ESM1-5_rainfall  \
0      1889-01-01             0.040427                1.814552
1      1889-01-02             0.073777                0.303965
2      1889-01-03             0.232656                0.019976
3      1889-01-04             0.911319               13.623777
4      1889-01-05             0.698013                0.021048
…             …                  …                     …
46015  2014-12-27             0.033748                0.123476
46016  2014-12-28             0.094198                2.645496
46017  2014-12-29             0.005964                3.041667
46018  2014-12-30             0.000028                1.131412
46019  2014-12-31             0.532747                2.370896

       AWI-ESM-1-1-LR_rainfall  BCC-CSM2-MR_rainfall  BCC-ESM1_rainfall  \
0                 3.557934e+01          4.268112e+00       1.107466e-03
1                 4.596520e+00          1.190141e+00       1.015323e-04
2                 5.927467e+00          1.003845e-09       1.760345e-05
3                 8.029624e+00          8.225225e-02       1.808932e-01
4                 2.132686e+00          2.496841e+00       4.708019e-09
…                          …                     …                  …
46015             1.451179e+00          3.852845e+01       2.061717e-03
46016             4.249335e+01          5.833801e-01       5.939502e-09
46017             2.898325e+00          9.359547e-02       2.000051e-08
46018             2.516381e-01          1.715028e-01       7.191735e-05
46019             1.047835e-13          4.437736e+00       2.863683e-01

       CMCC-CM2-HR4_rainfall  CMCC-CM2-SR5_rainfall  CMCC-ESM2_rainfall  \
```

```
0              1.141054e+01           3.322009e-08             2.668800
1              4.014984e+00           1.312700e+00             0.946211
2              9.660565e+00           9.103720e+00             0.431999
3              3.951528e+00           1.317160e+01             0.368693
4              2.766362e+00           1.822940e+01             0.339267
…                         …                      …                  …
46015          8.179260e-09           1.171263e-02             0.090786
46016          8.146937e-01           4.938899e-01             0.000000
46017          2.532205e-01           1.306046e+00             0.000002
46018          8.169252e-02           1.722262e-01             0.788577
46019          6.343592e+00           6.368303e-01             0.442130

       CanESM5_rainfall  …  MPI-ESM-1-2-HAM_rainfall  \
0              1.321215  …              4.244226e-13
1              2.788724  …              4.409552e+00
2              0.003672  …              2.269300e-01
3              0.013578  …              2.344586e-02
4              0.002468  …              4.270161e-13
…                   …  …                         …
46015         59.895053  …              4.726998e-13
46016          0.512632  …              4.609420e-13
46017         37.169669  …              2.016156e+01
46018          7.361246  …              9.420543e+00
46019          0.306608  …              1.031899e+01

       MPI-ESM1-2-HR_rainfall  MPI-ESM1-2-LR_rainfall  MRI-ESM2-0_rainfall  \
0                1.390174e-13            6.537884e-05         3.445495e-06
1                1.222283e-01            1.049131e-13         4.791993e-09
2                3.762301e-01            9.758706e-14         6.912302e-01
3                4.214019e-01            7.060915e-03         3.835721e-02
4                1.879692e-01            4.504985e+00         3.506923e-07
…                         …                      …                  …
46015            1.326889e-01            1.827857e+00         6.912632e-03
46016            1.644482e+00            7.242920e-01         2.836752e-03
46017            1.506439e+00            1.049481e-01         8.137182e+00
46018            6.242895e+00            1.245115e-01         9.305263e-03
46019            4.765813e+01            3.274323e-01         6.854985e-11

       NESM3_rainfall  NorESM2-LM_rainfall  NorESM2-MM_rainfall  \
0        1.576096e+01         4.759651e-05             2.451075
1        3.675510e-01         4.350863e-01             0.477231
2        1.562869e-01         9.561101e+00             0.023083
3        2.472226e-07         5.301038e-01             0.002699
4        1.949792e-13         1.460928e-10             0.001026
…                 …                    …                  …
46015    2.171327e-03         1.620489e+00             2.084252
46016    1.344768e+01         2.391159e+00             1.644527
```

```
46017    2.547820e+01        1.987695e-12            0.205036
46018    4.192948e+00        2.150346e+00            0.000017
46019    2.067847e+00        2.349716e+01            0.035319

       SAM0-UNICON_rainfall  TaiESM1_rainfall  observed_rainfall
0                  0.221324          2.257933           0.006612
1                  3.757179          2.287381           0.090422
2                  0.253357          1.199909           1.401452
3                  2.185454          2.106737          14.869798
4                  2.766507          1.763335           0.467628
…                    …                 …                 …
46015              0.868046         17.444923           0.037472
46016              0.782258          1.569647           0.158061
46017              2.140723          1.444630           0.025719
46018             29.714692          0.716019           0.729390
46019             59.724062          3.240185           0.008076

[46020 rows x 27 columns]
```

[4]:
```python
## Use your ML skills to get from step 1 to step 6
```

[5]:
```python
# Drop rows with na
df = df.dropna()
```

[6]:
```python
# Split the data into train (80%) and test (20%) portions with random_state=123

train, test = train_test_split(df, test_size=0.2, random_state=123)
```

[7]:
```python
# Carry out EDA of your choice on the train split

# Distribution of target label
(alt.Chart(train).mark_bar().encode(
    alt.X('observed_rainfall', bin=alt.Bin(maxbins=10), title='Observed␣
↪Rainfall (mm)'),
    y='count()'
))
```

[7]: alt.Chart(…)

[8]:
```python
# Observed rainfall over time
train['time'] = pd.to_datetime(train['time'])
(alt.Chart(train).mark_line().encode(
    x='time',
    y='observed_rainfall'
))
```

[8]: alt.Chart(…)

[9]:
```python
# Train ensemble machine learning model using RandomForestRegressor and␣
↪evaluate with metric of your choice (e.g., RMSE) by considering Observed as␣
↪the target column
y_train = train['observed_rainfall']
X_train = train.drop(['observed_rainfall', 'time'], axis=1)

m_rf = RandomForestRegressor()
res = cross_validate(m_rf, X_train, y_train, cv=5,␣
↪scoring='neg_root_mean_squared_error')
print(f"CV-train RMSE: {res['test_score'].mean()*-1}")
m_rf.fit(X_train, y_train)
```

```
CV-train RMSE: 8.330195567524267
```

[9]:
```
RandomForestRegressor()
```

[10]:
```python
m_rf.fit(X_train, y_train)
```

[10]:
```
RandomForestRegressor()
```

[11]:
```python
# Predict on test dataset
y_test = test['observed_rainfall']
X_test = test.drop(['observed_rainfall', 'time'], axis=1)

test['Ensemble'] = m_rf.predict(X_test)
```

[12]:
```python
# Calculate RMSE for all models
test_wide = test.drop(['time', 'observed_rainfall'], axis=1)
test_wide = pd.melt(test_wide, var_name='model', value_name='rainfall')

test_result = {}

for model in test_wide['model'].unique():
    pred = test_wide.query('model == @model')['rainfall']
    test_result[model] = mean_squared_error(y_test, pred, squared=False)
```

[13]:
```python
pd.DataFrame([test_result]).T.rename(columns={0: "RMSE"}).sort_values('RMSE')
```

[13]:

	RMSE
Ensemble	8.846681
KIOST-ESM_rainfall	9.600480
FGOALS-g3_rainfall	9.687788
MRI-ESM2-0_rainfall	9.922795
MPI-ESM1-2-HR_rainfall	9.969823
NESM3_rainfall	9.978137
MPI-ESM1-2-LR_rainfall	10.260886
NorESM2-LM_rainfall	10.410145

```
EC-Earth3-Veg-LR_rainfall  10.453606
GFDL-CM4_rainfall          10.511682
BCC-ESM1_rainfall          10.615578
CMCC-CM2-HR4_rainfall      10.643204
ACCESS-ESM1-5_rainfall     10.695305
BCC-CSM2-MR_rainfall       10.761381
MPI-ESM-1-2-HAM_rainfall   10.932004
NorESM2-MM_rainfall        10.939740
AWI-ESM-1-1-LR_rainfall    10.996616
ACCESS-CM2_rainfall        11.038999
CanESM5_rainfall           11.151318
CMCC-ESM2_rainfall         11.246493
MIROC6_rainfall            11.352976
INM-CM4-8_rainfall         11.451635
CMCC-CM2-SR5_rainfall      11.480614
TaiESM1_rainfall           11.528083
SAM0-UNICON_rainfall       11.678749
INM-CM5-0_rainfall         12.250223
```

> Discuss your results. Are you getting better results with ensemble models compared to the individual climate models

Yes, we are getting better results with ensemble model compared to individual climate model. This is because by ensembling different models together, we are able to cancel out the errors made by different models, and combine different mapping function learnt by each individual climate model.

2.2 Part 2:

2.2.1 Preparation for deploying model next week

NOTE: Complete task 4 from the milestone3 before coming here

We've found the best hyperparameter settings with MLlib (from the task 4 from milestone3), here we then use the same hyperparameters to train a scikit-learn model.

[14]:
```python
model = RandomForestRegressor(n_estimators=100, max_depth=5, bootstrap=False)
model.fit(X_train, y_train)
```

[14]:
```
RandomForestRegressor(bootstrap=False, max_depth=5)
```

[15]:
```python
print(f"Train RMSE: {mean_squared_error(y_train, model.predict(X_train),␣
 ↪squared=False):.2f}")
print(f" Test RMSE: {mean_squared_error(y_test, model.predict(X_test),␣
 ↪squared=False):.2f}")
```

```
Train RMSE: 7.91
 Test RMSE: 8.71
```

[16]:
```python
# ready to deploy
dump(model, "model.joblib")
```

[16]: ['model.joblib']

Upload model.joblib to s3 under output folder. You choose how you want to upload it (using CLI, SDK, or web console).