Carbon Prediction with Machine Learning

Importing libraries

In [1]:
```
# Import libraries
import pandas as pd
import numpy as np
import random
import os
from tqdm.notebook import tqdm

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
pd.options.display.float_format = '{:.5f}'.format
pd.options.display.max_rows = None

%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# Set seed for reproducibility
SEED = 2023
random.seed(SEED)
np.random.seed(SEED)
```

Loading and previewing data

In [3]:
```
DATA_PATH = '/input/playground-series-s3e20'
# Load files
train = pd.read_csv(os.path.join(DATA_PATH, 'train.csv'))
test = pd.read_csv(os.path.join(DATA_PATH, 'test.csv'))

# Preview train dataset
train.head()
```

Out[3]:

	ID_LAT_LON_YEAR_WEEK	latitude	longitude	year	week_no	SulphurDioxide_SO2_column_number_density	SulphurDioxide_SO2_column_nun
0	ID_-0.510_29.290_2019_00	-0.51000	29.29000	2019	0	-0.00011	
1	ID_-0.510_29.290_2019_01	-0.51000	29.29000	2019	1	0.00002	
2	ID_-0.510_29.290_2019_02	-0.51000	29.29000	2019	2	0.00051	
3	ID_-0.510_29.290_2019_03	-0.51000	29.29000	2019	3	NaN	
4	ID_-0.510_29.290_2019_04	-0.51000	29.29000	2019	4	-0.00008	

5 rows × 76 columns

In [4]:
```
# Preview test dataset
test.head()
```

Out[4]:

	ID_LAT_LON_YEAR_WEEK	latitude	longitude	year	week_no	SulphurDioxide_SO2_column_number_density	SulphurDioxide_SO2_column_nun
0	ID_-0.510_29.290_2022_00	-0.51000	29.29000	2022	0	NaN	
1	ID_-0.510_29.290_2022_01	-0.51000	29.29000	2022	1	0.00046	
2	ID_-0.510_29.290_2022_02	-0.51000	29.29000	2022	2	0.00016	
3	ID_-0.510_29.290_2022_03	-0.51000	29.29000	2022	3	0.00035	
4	ID_-0.510_29.290_2022_04	-0.51000	29.29000	2022	4	-0.00032	

5 rows × 75 columns

In [5]:
```
# Preview sample submission file
samplesubmission.head()
```

Out[5]:

	ID_LAT_LON_YEAR_WEEK	emission
0	ID_-0.510_29.290_2022_00	81.94000
1	ID_-0.510_29.290_2022_01	81.94000
2	ID_-0.510_29.290_2022_02	81.94000
3	ID_-0.510_29.290_2022_03	81.94000
4	ID_-0.510_29.290_2022_04	81.94000

In [6]:
```
# Check size and shape of datasets
train.shape, test.shape, samplesubmission.shape
```

```
Out[6]: ((79023, 76), (24353, 75), (24353, 2))
```

```
In [7]: # Train to test sets ratio
        (test.shape[0]) / (train.shape[0] + test.shape[0])
```

```
Out[7]: 0.23557692307692307
```

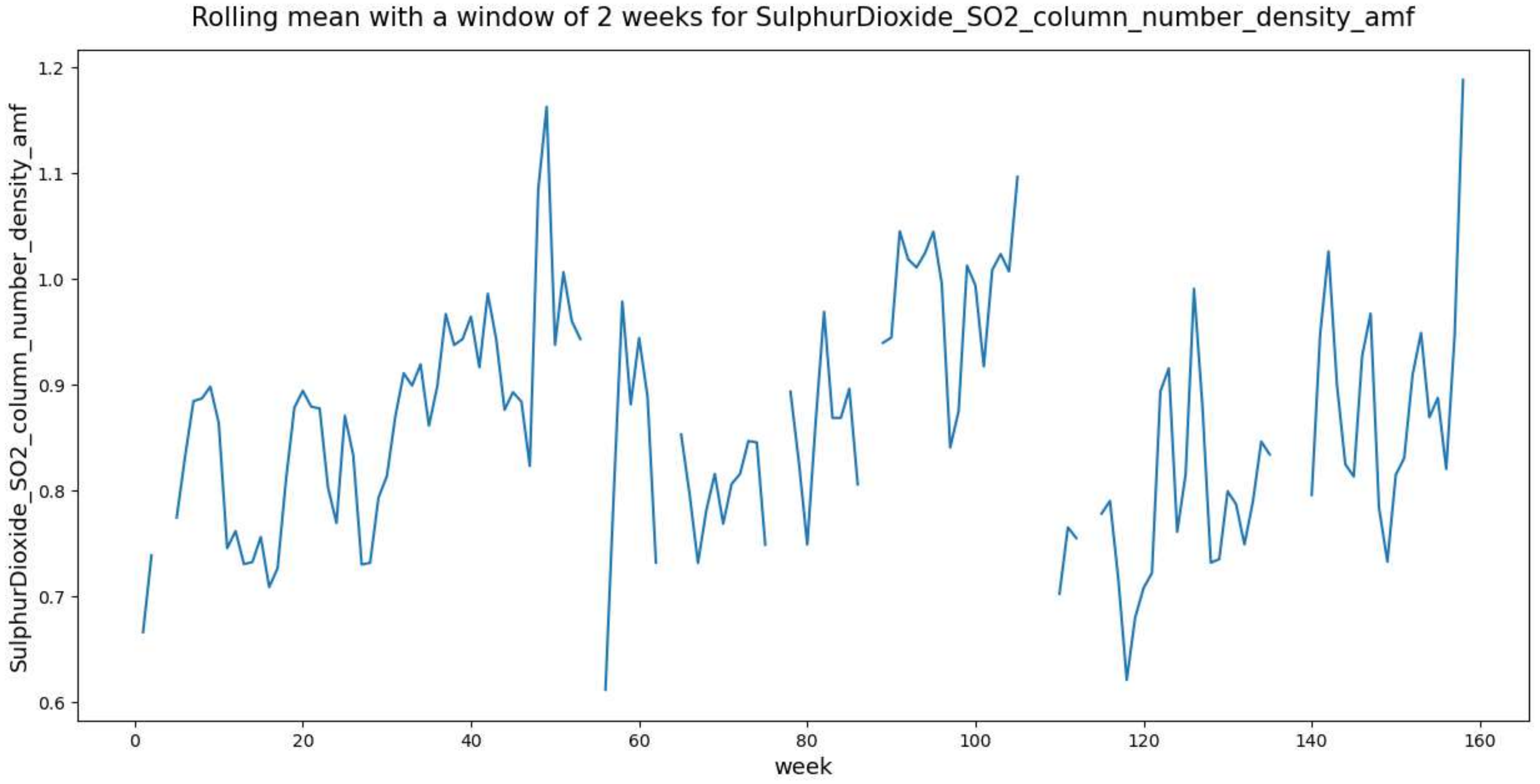
Feature engineering

```
In [8]: # First we create a unique location from lat lon
        train['location'] = [str(x) + '_' + str(y) for x, y in zip(train.latitude, train.longitude)]

        # Filter based on one location
        example_loc = train[train.location == '-0.51_29.29']

        # Calculate rolling mean for SulphurDioxide_SO2_column_number_density_amf with a window of 2 weeks
        rolling_mean = example_loc['SulphurDioxide_SO2_column_number_density_amf'].rolling(window = 2).mean()

        # Visualise rolling mean
        plt.figure(figsize = (15, 7))
        rolling_mean.plot()
        plt.title('Rolling mean with a window of 2 weeks for SulphurDioxide_SO2_column_number_density_amf', y = 1.02, fontsize
        plt.xlabel('week', y = 1.05, fontsize = 13)
        plt.ylabel('SulphurDioxide_SO2_column_number_density_amf', x = 1.05, fontsize = 13)
        plt.show()
```



- With more research and domain knowledge generate useful features that can improve your model performance

  Other examples of feature engineering:

  - Creating cluster regions
  - Interactions between different pollutatnts - ratios, additions,subtractions...
  - Time series features

```
In [9]: # Generate the above feature - rolling mean for all locations for both the train and test

        # Feature engineering train
        train_roll_mean = train.sort_values(by = ['location', 'year', 'week_no']).groupby(['location'])[train.columns[5:].tolis
        train_roll_mean.drop(['level_1', 'emission', 'location'], axis = 1, inplace = True)
        train_roll_mean.columns = [col + '_roll_mean' for col in train_roll_mean.columns]

        # Feature engineering test
        test.latitude, test.longitude = round(test.latitude, 2), round(test.longitude, 2)
        test['location'] = [str(x) + '_' + str(y) for x, y in zip(test.latitude, test.longitude)]
        test_roll_mean = test.sort_values(by = ['location', 'year', 'week_no']).groupby(['location'])[test.columns[5:].tolist()
        test_roll_mean.drop(['level_1', 'location'], axis = 1, inplace = True)
        test_roll_mean.columns =  [col + '_roll_mean' for col in test_roll_mean.columns]
        test_roll_mean.head()
```

Out[9]:

	SulphurDioxide_SO2_column_number_density_roll_mean	SulphurDioxide_SO2_column_number_density_amf_roll_mean	SulphurDioxide_SO2_slant
0	NaN	NaN	
1	NaN	NaN	
2	0.00031	0.64814	
3	0.00026	0.65101	
4	0.00002	0.63872	

5 rows × 70 columns

In [10]:

```python
# Merge engineered features with train and test set

#Train
train_eng = train.sort_values(by = ['location', 'year', 'week_no'], ignore_index = True).merge(train_roll_mean, how = '
                                                                                        left_index=True, right_i

# Test
test_eng = test.sort_values(by = ['location', 'year', 'week_no'], ignore_index = True).merge(test_roll_mean, how = 'lef
                                                                                        left_index=True, right_i

# Preview engineered test set
test_eng.head()
```

Out[10]:

	ID_LAT_LON_YEAR_WEEK	latitude	longitude	year	week_no	SulphurDioxide_SO2_column_number_density	SulphurDioxide_SO2_column_nun
0	ID_-0.510_29.290_2022_00	-0.51000	29.29000	2022	0	NaN	
1	ID_-0.510_29.290_2022_01	-0.51000	29.29000	2022	1	0.00046	
2	ID_-0.510_29.290_2022_02	-0.51000	29.29000	2022	2	0.00016	
3	ID_-0.510_29.290_2022_03	-0.51000	29.29000	2022	3	0.00035	
4	ID_-0.510_29.290_2022_04	-0.51000	29.29000	2022	4	-0.00032	

5 rows × 146 columns

Modelling

In [11]:

```python
# Selecting the independent variables and the target variable

X = train_eng.drop(['ID_LAT_LON_YEAR_WEEK', 'location', 'emission'], axis = 1).fillna(0)
y = train_eng.emission

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = SEED)

# Instantiating the model
clf = RandomForestRegressor(random_state = SEED, n_jobs=-1)
clf.fit(X_train, y_train)

# Making predictions
y_pred = clf.predict(X_test)

# Measuring the accuracy of the model
print(f'RMSE Score: {mean_squared_error(y_test, y_pred, squared=False)}') # 27.46875858227988
```

```
RMSE Score: 27.46875858227988
```

In [12]:

```python
X_test.head()
```

Out[12]:

	latitude	longitude	year	week_no	SulphurDioxide_SO2_column_number_density	SulphurDioxide_SO2_column_number_density_amf	Sul
42962	-1.96800	30.93200	2019	32	-0.00031	0.71053	
20489	-1.32700	30.97300	2021	31	-0.00005	0.83240	
49300	-2.17100	28.62900	2019	10	0.00008	0.82463	
13289	-1.11700	29.88300	2020	39	0.00000	0.00000	
31375	-1.64100	31.25900	2019	52	0.00004	0.75535	

5 rows × 144 columns

In [13]:

```python
# Analyse predictions
pred_errors = X_test.copy()
pred_errors['emission'] = y_test
pred_errors['prediction'] = y_pred
pred_errors['error'] = abs(pred_errors.prediction - pred_errors.emission)
pred_errors = pred_errors[['latitude',  'longitude',    'year', 'week_no', 'emission', 'prediction', 'error']]
pred_errors.sort_values(by = 'error', ascending = False, inplace = True)
pred_errors.head()
```

Out[13]:

	latitude	longitude	year	week_no	emission	prediction	error
46437	-2.07900	29.32100	2019	9	1044.48450	2923.18510	1878.70060
46490	-2.07900	29.32100	2020	9	1011.02600	2795.15162	1784.12562
56674	-2.37800	29.22200	2020	17	1502.66770	2079.50193	576.83423
56679	-2.37800	29.22200	2020	22	1689.61380	2223.95643	534.34263
56671	-2.37800	29.22200	2020	14	1777.71030	2244.19961	466.48931

In [14]:
```
pred_errors.tail()
```

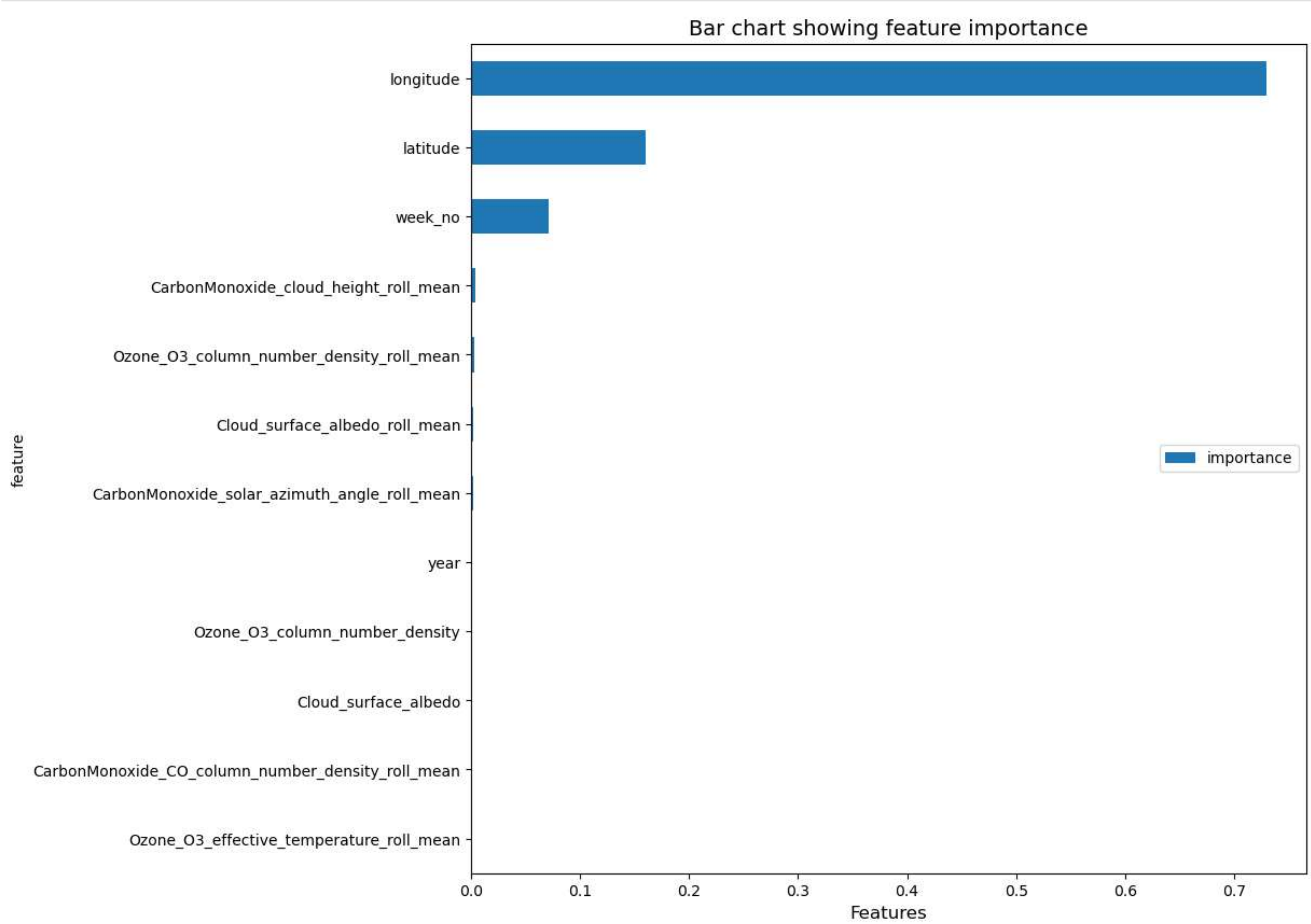
Out[14]:

	latitude	longitude	year	week_no	emission	prediction	error
33233	-1.71200	28.68800	2019	2	0.00000	0.00000	0.00000
70859	-2.84100	29.15900	2020	51	0.00000	0.00000	0.00000
36595	-1.83300	28.46700	2019	25	0.00000	0.00000	0.00000
26598	-1.50500	30.99500	2019	45	0.00000	0.00000	0.00000
71947	-2.85900	29.04100	2020	26	0.00000	0.00000	0.00000

In [15]:
```
train.emission.describe()
```

Out[15]:
```
count   79023.00000
mean       81.94055
std       144.29965
min         0.00000
25%         9.79800
50%        45.59345
75%       109.54959
max      3167.76800
Name: emission, dtype: float64
```

In [16]:
```
# Feature importance
impo_df = pd.DataFrame({'feature': X.columns, 'importance': clf.feature_importances_}).set_index('feature').sort_values
impo_df = impo_df[:12].sort_values(by = 'importance', ascending = True)
impo_df.plot(kind = 'barh', figsize = (10, 10))
plt.legend(loc = 'center right')
plt.title('Bar chart showing feature importance', fontsize = 14)
plt.xlabel('Features', fontsize = 12)
plt.show()
```



Making predictions of the test set and creating a submission file

In [17]:
```
# Make prediction on the test set
test_df = test_eng.drop(['ID_LAT_LON_YEAR_WEEK', 'location'], axis = 1).fillna(0)
predictions = clf.predict(test_df)

# # Create a submission file
sub_file = pd.DataFrame({'ID_LAT_LON_YEAR_WEEK': test_eng.ID_LAT_LON_YEAR_WEEK, 'emission': predictions})
sub_file.head()
```

Out[17]:

	ID_LAT_LON_YEAR_WEEK	emission
0	ID_-0.510_29.290_2022_00	3.64300
1	ID_-0.510_29.290_2022_01	4.19987
2	ID_-0.510_29.290_2022_02	4.27457
3	ID_-0.510_29.290_2022_03	4.36166
4	ID_-0.510_29.290_2022_04	4.08831