Air Pollutant Concentration Prediction over Ahmedabad Using Machine Learning

CSE523 - Machine Learning
Winter Semester 2023
Weekly Report - 8/4/2023

Yash Dahima - AU2129002

The task of *model development* using kernel-based models has been performed this week.

Kernel Ridge Model:

Ridge Regression model was developed using a periodic kernel with random search of the best parameters. 100 iterations were performed to choose the best combination of alpha, kernel length scale, and kernel periodicity. The model was fit with the dataset and evaluated using different metrics. Randomized Search provided the best values of alpha = 771.7, kernel length scale = 32.0, and kernel periodicity = 0.22 out of 100 iterations.

```python
# -*- coding: utf-8 -*-
"""

Kernel Ridge Regression

"""

import pandas as pd, numpy as np

from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.gaussian_process.kernels import ExpSineSquared
from sklearn.kernel_ridge import KernelRidge
from sklearn.utils.fixes import loguniform

from sklearn.metrics import explained_variance_score, mean_absolute_error, mean_squared_error,
r2_score, mean_absolute_percentage_error


# Load the dataset
df = pd.read_excel('C:/Users/Yash Dahima/PhD/Course Work/ML/Project/AQI/Datasets/data4.xlsx')
df['datetime'] = pd.to_datetime(df['datetime'])
df = df.set_index('datetime')

# Separate the target variable from the features
X = df.drop('pm2p5', axis=1)
y = df['pm2p5']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)

kernel_ridge = KernelRidge(kernel=ExpSineSquared())

param_distributions = {
    "alpha": loguniform(1e-2, 1e3),
    "kernel__length_scale": loguniform(1e-2, 1e2),
    "kernel__periodicity": loguniform(1e-2, 1e2),
}
```

```python
kernel_ridge_tuned = RandomizedSearchCV(
    kernel_ridge,
    param_distributions=param_distributions,
    n_iter=100,
    random_state=0,
)


# Train the model
kernel_ridge_tuned.fit(X_train, y_train)

# Evaluate the model on the testing set
y_pred = kernel_ridge_tuned.predict(X_test)

# Compute evaluation metrics
#coeff = model.coef_
evs = explained_variance_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
mape = mean_absolute_percentage_error(y_test, y_pred)*100
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)
```

Gaussian Process Regressor:

The Gaussian Process Regressor model was developed using a periodic kernel to address daily and seasonal periodicity, and quadratic kernel to address trend in the dataset.

```python
# -*- coding: utf-8 -*-
"""

Gaussian Process Regressor

"""

import pandas as pd, numpy as np, time
from scipy.stats import pearsonr, spearmanr

from sklearn.model_selection import train_test_split
from sklearn.gaussian_process.kernels import ExpSineSquared, WhiteKernel, RBF, RationalQuadratic
from sklearn.gaussian_process import GaussianProcessRegressor

from sklearn.metrics import explained_variance_score, mean_absolute_error, mean_squared_error,
r2_score, mean_absolute_percentage_error


# Load the dataset
df = pd.read_excel('C:/Users/Yash Dahima/PhD/Course Work/ML/Project/AQI/Datasets/data4.xlsx')
df['datetime'] = pd.to_datetime(df['datetime'])
df = df.set_index('datetime')
```

```python
# Separate the target variable from the features
X = df.drop('pm2p5', axis=1)
y = df['pm2p5']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)

# Define the kernels
kernel_daily = ExpSineSquared(length_scale=1.0, periodicity=1.0)
kernel_seasonal = ExpSineSquared(length_scale=1.0, periodicity=365.0)
kernel_trend = RationalQuadratic(length_scale=1.0, alpha=1.0)

# Combine the kernels using addition and multiplication
kernel = RBF(length_scale=1.0) * (kernel_daily + kernel_seasonal) + kernel_trend

# Initialize the GaussianProcessRegressor model
gaussian_process = GaussianProcessRegressor(kernel=kernel)

# Fit the model to the training data
start_time = time.time()
gaussian_process.fit(X_train, y_train)
print(f"Time for GaussianProcessRegressor fitting: {time.time() - start_time:.3f} seconds")

# Evaluate the model on the testing set
y_pred = gaussian_process.predict(X_test)

# Compute evaluation metrics
#coeff = model.coef_
evs = explained_variance_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
mape = mean_absolute_percentage_error(y_test, y_pred)*100
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)
```

Important model performance evaluation metrics are shown in the table below:

Models	Kernel Ridge	Gaussian Process
Mean Absolute Error	40.4	32.4
Mean Absolute Percentage Error	47.4	44.5
Root Mean Squared Error	53.9	40.5
R2 Score	- 0.2	0.1

As we can see, the model performances are not satisfactory, they are still not able to capture the pattern in the data. Hence, the models such as Random Forest, SVR will be developed in the next week.