Air Pollutant Concentration Prediction over Ahmedabad Using Machine Learning

CSE523 - Machine Learning
Winter Semester 2023
Weekly Report - 15/4/2023

Yash Dahima - AU2129002

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

Ensemble Models:

Four models extra-trees, random forest, gradient boosting, and histogram-based gradient boosting were fit on the dataset using scikit-learn. They were run with the same random state and 10,000 number of estimators for the inter-comparison. Extra-trees performed best in this run. The models were again run with different parameters and gradient boosting performed best this time. There wasn't a big difference in the performances of these models.

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

Ensemble Models

"""

import pandas as pd, numpy as np, time
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import ExtraTreesRegressor, GradientBoostingRegressor, HistGradientBoostingRegressor,
RandomForestRegressor

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)


# Create a list of models to evaluate
models = [ExtraTreesRegressor(n_estimators=10000, random_state=0, n_jobs=-1, criterion='friedman_mse',
min_samples_split=20, min_samples_leaf=5, max_features=0.3),
          RandomForestRegressor(n_estimators=10000, random_state=0, n_jobs=-1, criterion='friedman_mse',
min_samples_split=20, min_samples_leaf=5, max_features=0.3),
          GradientBoostingRegressor(n_estimators=10000, random_state=0, loss='huber', learning_rate=0.01,
subsample=0.9, criterion='friedman_mse', min_samples_split=20, min_samples_leaf=5, max_depth=None,
max_features=0.3),
          HistGradientBoostingRegressor(max_iter=10000, random_state=0, max_leaf_nodes=None,
learning_rate=0.01, min_samples_leaf=5)
          ]
```

```python
# Create an empty dataframe to store the evaluation metrics
metrics_df = pd.DataFrame(columns=["Model", "evs", "mae", "mape", "rmse", "r^2 Score"])


# Evaluate each model using a for loop
for model in models:

    start_time = time.time()

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

    print(f"Time for fitting: {(time.time() - start_time)/60:.3f} minutes")

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

    # Compute evaluation metrics
    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)

    # Append evaluation metrics to the dataframe
    metrics_df = pd.concat([metrics_df, pd.DataFrame({"Model": [type(model).__name__],
                                                       "evs": [evs],
                                                       "mae": [mae],
                                                       "mape": [mape],
                                                       "rmse": [rmse],
                                                       "r^2 Score": [r2]})],
                           ignore_index=True)

```

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

Models	**Extra-trees**	**Random Forest**	**Gradient Boosting**	**Hist-Gradient Boosting**
Mean Absolute Error	13.64	13.88	14.85	14.04
Mean Absolute Percentage Error	22.68	22.91	24.02	22.83
Root Mean Squared Error	18.03	18.33	19.71	18.65
R2 Score	0.82	0.81	0.78	0.80

As we can see, the model performances are far better than the previous one, they seem to be able to capture the pattern in the data.