KNN regressor

2024-11-27

Contents

In this report, I perform K Nearest Neighbour (KNN) model to predict saturation vapour pressure. KNN predicts the value of the target variable based on the average of the values of its k-nearest neighbors in the training set. Since KNN does not accept missing values in the dataset, I run the model on complete-case train data, and imputed train data, and compare their performance.

Complete_case train data

Validation set method

```
import pandas as pd
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score, KFold


# read complete-case train data
train = pd.read_csv('../data/train_complete_cases.csv')


# store target variable and features
X = train.iloc[:, 2:]
y = train['log_pSat_Pa']
# split data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=100)
# define the model
knn = KNeighborsRegressor(n_neighbors=5)
# fit the model
knn.fit(X_train, y_train)
# predict y in test data
```

```
## KNeighborsRegressor()
```

```
y_pred = knn.predict(X_test)
# calculate mse
mse_vali = round(mean_squared_error(y_test, y_pred), 3)
print(f'MSE obtained from validation set method: {mse_vali}')
```

```
## MSE obtained from validation set method: 5.541
```

Cross-validation method

```
# store target variable and features
X_train = train.iloc[:, 2:]
y_train = train['log_pSat_Pa']
# define the model
model = KNeighborsRegressor(n_neighbors=5)
# apply cross-validation
k = 10
kf = KFold(n_splits=k, shuffle=True, random_state=100)
fold_scores = -cross_val_score(model, X_train, y_train, cv=kf, scoring='neg_mean_squared_error')
mse_cv = round(fold_scores.mean(), 3)
print(f'MSE obtained from cross-validation method: {mse_cv}')
```

```
## MSE obtained from cross-validation method: 5.419
```

Imputed train data

Validation set method

```
# read complete-case train data
train = pd.read_csv('../data/train_imputed.csv')

# store target variable and features
X = train.iloc[:, 2:]
y = train['log_pSat_Pa']
# split data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=100)
# define the model
knn = KNeighborsRegressor(n_neighbors=5)
# fit the model
knn.fit(X_train, y_train)
# predict y in test data
```

```
## KNeighborsRegressor()
```

```
y_pred = knn.predict(X_test)
# calculate mse
mse_vali = round(mean_squared_error(y_test, y_pred), 3)
print(f'MSE obtained from validation set method: {mse_vali}')
```

```
## MSE obtained from validation set method: 5.617
```

Cross-validation method

```
# store target variable and features
X_train = train.iloc[:, 2:]
y_train = train['log_pSat_Pa']
# define the model
model = KNeighborsRegressor(n_neighbors=5)
# apply cross-validation
k = 10
kf = KFold(n_splits=k, shuffle=True, random_state=100)
fold_scores = -cross_val_score(model, X_train, y_train, cv=kf, scoring='neg_mean_squared_error')
mse_cv = round(fold_scores.mean(), 3)
print(f'MSE obtained from cross-validation method: {mse_cv}')
```

```
## MSE obtained from cross-validation method: 5.428
```

The results show that the cross-validation method performs better compared to the validation set approach. The MSE for complete-case data (5.543) is lower than for imputed data (5.617) when using the validation set method. When applying cross-validation, the differences in MSE between the complete-case data (5.420) and the imputed data (5.427) are somewhat canceled out.

Future work:

- feature selection
- applying different loss function
- evaluating the model performance on train data