EVCS density clustering ML model Improvement Summary Report

Previous issues identified from the revaluation and their improvements.

1) Cleaned data are suitable for supervise machine learning model, however the data have been further converted using “time series data to supervised data” function, this might add unnecessary computation loads by increasing feature dimensions, and converted data might not well represent the original data.

Improvements:

a. Using cleaned data without further data transformation

```
]: pred_df
]:
```

	Postal Code	Plugs_Type2	Plugs_Three_Phase	Plugs_CHAdeMO	Plugs_CCS/SAE	Plugs_Tesla	Plugs 1
0	0	1.0	0.0	0.0	0.0	0.0	
1	92	0.0	0.0	0.0	0.0	0.0	
2	2	0.0	0.0	0.0	0.0	2.0	
3	146	0.0	0.0	4.0	4.0	0.0	
4	123	0.0	0.0	2.0	2.0	0.0	
...	...	...	...	...	...	...	
386	14	2.0	0.0	0.0	0.0	0.0	
387	2	0.0	0.0	0.0	0.0	0.0	
388	30	0.0	0.0	0.0	0.0	0.0	
389	96	1.0	0.0	0.0	0.0	0.0	
390	146	0.0	0.0	0.0	0.0	0.0	

b. Keeping the “Postal Code” feature for data training since it has high correlation to target feature Clusters.

```
correlation
```

Clusters	1.000000
Postal Code	0.397551
Parks	0.388432
Restaurants	0.349416
Nearby EVStations	0.324928
Supermarkets	0.278466
Plugs_J-1772	0.224222
Power 1	0.221292
Malls	0.208359
charging_stations	0.168188
Plugs_Caravan_Mains_Socket	0.072910
Plugs_CHAdeMO	0.060516
Plugs_wall_AU/NZ	0.055808
Plugs_Type2	0.047321
Hospitals	0.038234
Plugs_Three_Phase	0.031584
Plugs_CCS/SAE	0.027776
Plugs_Tesla	0.012261

c. Reduce numbers of predictor features from previous 17 to 9, to reduce computational complexity and Nosie.

	Postal Code	Parks	Restaurants	Nearby EVStations	Supermarkets	Power 1	Malls	charging_stations	Hospitals	C
0	2714	0	0	0	0	25	0	0.0	0	
1	3205	18	20	15	20	24	4	21.0	1	
2	3000	18	20	20	20	24	20	2.0	2	
3	3943	5	20	1	6	17	0	8.0	0	
4	3757	9	20	1	1	17	0	4.0	0	
...	...	...	...	...	...	...	...	...	...	
386	3023	10	20	1	7	24	1	2.0	0	
387	3000	17	20	16	20	24	20	3.0	7	
388	3053	18	20	16	20	24	20	2.0	11	
389	3216	5	15	2	3	8	0	1.0	0	
390	3943	7	20	1	6	24	0	1.0	0	

Barry Chen

2) The feature data including for training did not remove the target variable data, the target variable data also fit in machine learning model for training, see feature 3 below. This would not able to train the model properly which will cause model overfitting, since the model already expecting there are 100% correlation between column 35 of the feature data and the target data.

```
]: # Preprocessing
   # Splitting data into features and target variable

   X = data # Features
   y = data[35] # Target variable

]: print(X)
   print(X.shape)

        0    1    2    3    4    5    6    7    8    9   ...   26   27  \
0     0.0  1.0  0.0  0.0  0.0  0.0  5.0  0.0  0.0  25.0  ...  21.0  24.0
1    92.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0  21.0  24.0  ...   0.0  24.0
2     2.0  0.0  0.0  0.0  0.0  2.0  1.0  0.0  0.0  24.0  ...   0.0  17.0
3   146.0  0.0  0.0  4.0  4.0  0.0  1.0  0.0  0.0  17.0  ...   0.0  17.0
4   123.0  0.0  0.0  2.0  2.0  0.0  1.0  0.0  0.0  17.0  ...   1.0  24.0
..    ...  ...  ...  ...  ...  ...  ...  ...  ...   ...  ...   ...   ...
385 131.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0  1.0  24.0  ...   0.0  24.0
386  14.0  2.0  0.0  0.0  0.0  0.0  1.0  0.0  0.0  24.0  ...   3.0  24.0
387   2.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0  3.0  24.0  ...   0.0  24.0
388  30.0  0.0  0.0  0.0  0.0  0.0  3.0  0.0  0.0  24.0  ...   0.0   8.0
389  96.0  1.0  0.0  0.0  0.0  0.0  1.0  0.0  0.0   8.0  ...   1.0  24.0

       28    29   30    31    32    33    34   35
0    21.0  15.0  1.0  18.0  20.0   4.0  20.0  1.0
1     2.0  20.0  2.0  18.0  20.0  20.0  20.0  0.0
2     8.0   1.0  0.0   5.0  20.0   0.0   6.0  1.0
3     4.0   1.0  0.0   9.0  20.0   0.0   1.0  3.0
4     1.0   1.0  0.0   9.0   3.0   0.0   1.0  1.0
..    ...   ...  ...   ...   ...   ...   ...  ...
385   2.0   1.0  0.0  10.0  20.0   1.0   7.0  0.0
386   3.0  16.0  7.0  17.0  20.0  20.0  20.0  2.0
387   2.0  16.0 11.0  18.0  20.0  20.0  20.0  0.0
388   1.0   2.0  0.0   5.0  15.0   0.0   3.0  1.0
389   1.0   1.0  0.0   7.0  20.0   0.0   6.0  1.0

[390 rows x 35 columns]
(390, 35)

]: print(y)
   print(y.shape)

0      1.0
1      0.0
2      1.0
3      3.0
4      1.0
      ...
385    0.0
386    2.0
387    0.0
388    1.0
389    1.0
Name: 35, Length: 390, dtype: float64
(390,)
```

Feature (3): Variable Data set for training

The machine learning model accuracy drop significantly after removing the target variable from the feature variable for model training. The best accuracy is 0.5325 from random forest machine learning model.

```
[87]: print(tabulate(table, headers= 'firstrow', floatfmt=".4f" ,tablefmt =
```

Classifier	Base Accuracy	Grid Accuracy
Random Forest	0.9610	0.9351
KNN	0.5584	0.4805
XgBoost	1.0000	0.9610
SVM	0.4026	0.6753
Logistic Regression	0.8052	0.8701

Feature (4a): Accuracies from original Models

```
In [87]: print(tabulate(table, headers= 'firstrow', floatfmt=".4f" ,tablefmt = 'fancy_gric
```

Classifier	Base Accuracy	Grid Accuracy
Random Forest	0.5195	0.5325
KNN	0.5195	0.4545
XgBoost	0.5455	0.4805
SVM	0.3896	0.5195
Logistic Regression	0.5974	0.6104

Feature (4b): Accuracies of Models after dropping target variable.

Improvement: isolate target data from predictor data then retrain the model.

```
print(X.shape)

     Postal Code  Parks  Restaurants  Nearby EVStations  Supermarkets  \
0           2714      0            0                  0             0
1           3205     18           20                 15            20
2           3000     18           20                 20            20
3           3943      5           20                  1             6
4           3757      9           20                  1             1
..           ...    ...          ...                ...           ...
386         3023     10           20                  1             7
387         3000     17           20                 16            20
388         3053     18           20                 16            20
389         3216      5           15                  2             3
390         3943      7           20                  1             6

     Power 1  Malls  charging_stations  Hospitals
0         25      0                0.0          0
1         24      4               21.0          1
2         24     20                2.0          2
3         17      0                8.0          0
4         17      0                4.0          0
..       ...    ...                ...        ...
386       24      1                2.0          0
387       24     20                3.0          7
388       24     20                2.0         11
389        8      0                1.0          0
390       24      0                1.0          0

[391 rows x 9 columns]
(391, 9)

]: print(y)
   print(y.shape)

0      0
1      0
2      0
3      2
4      2
```

3) Other improvements:

a. Improvement model generalization by split training set into train/validation set, and only train on the validation set, then test on the test set.

```
36]: from sklearn.model_selection import train_test_split

     # Split dataset into training set and test set
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)

37]: # Check the shape of all of these
     print("X_train shape is : ", X_train.shape)
     print("X_test shape  is : ", X_test.shape)
     print("y_train shape is : ", y_train.shape)
     print("y_test shape is  : ", y_test.shape)

     X_train shape is :  (273, 9)
     X_test shape  is :  (118, 9)
     y_train shape is :  (273,)
     y_test shape is  :  (118,)

38]: #split into train/validation set
     M_train, M_test, N_train, N_test=train_test_split(X_train, y_train, test_size=0.3, random_state
```

b. Add random_state to each model to remove randomness on each run.

1. Random Forest

```
]: # Best Random Forest Model

   rfclf = RandomForestClassifier(random_state=1)
   rfclf.fit(M_train,N_train)

]:      RandomForestClassifier
   RandomForestClassifier(random_state=1)

]: # Predicted Outcome
   n_pred = rfclf.predict(M_test)
```

c. Create Print score function to improve code efficiency.

```
]: def printScore(testData,predictData):
       print(confusion_matrix(testData,predictData))
       print(classification_report(testData,predictData))

       # Model Accuracy: how often is the classifier correct?
       base_accuracy_rf = metrics.accuracy_score(testData,predictData)
       print("Accuracy:", base_accuracy_rf)

       # Model Precision: what percentage of positive tuples are labeled as such
       print("Precision:",metrics.precision_score(testData,predictData, average=

       # Model Recall: what percentage of positive tuples are labelled as such?
       print("Recall:",metrics.recall_score(testData,predictData, average='weigh

       #Calculate F1 Score
       print("F1 Score:",metrics.f1_score(testData,predictData, average='weighte

       #Calculate Mean Absolute Error
       print("Mean Absolute Error:",metrics.mean_absolute_error(testData,predictl
```

d. Cross- validation applied to generalize model.

```
print("10 fold cross validation result")
from sklearn.model_selection import cross_val_score
scores = cross_val_score(rfclf, X_train, y_train, cv=10)
print("%0.2f accuracy with a standard deviation of %0.2f" % (scores.mean(), scores.std()))
cvScore_rf=scores.mean()
```

```
10 fold cross validation result
0.97 accuracy with a standard deviation of 0.03
```

e. Only train on validation set then predict with test set.

```
Random Forest on validation set
[[55  0  0  0]
 [ 0  1  0  0]
 [ 2  0 16  0]
 [ 0  0  0  8]]
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        55
           1       1.00      1.00      1.00         1
           2       1.00      0.89      0.94        18
           3       1.00      1.00      1.00         8

    accuracy                           0.98        82
   macro avg       0.99      0.97      0.98        82
weighted avg       0.98      0.98      0.98        82

Accuracy: 0.975609756097561
Precision: 0.9764655541292255
Recall: 0.975609756097561
F1 Score: 0.9751101660176267
Mean Absolute Error: 0.04878048780487805
```

```
Random Forest on test set
[[76  0  0  0]
 [ 3  8  0  0]
 [ 0  0 23  0]
 [ 0  0  1  7]]
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        76
           1       1.00      0.73      0.84        11
           2       0.96      1.00      0.98        23
           3       1.00      0.88      0.93         8

    accuracy                           0.97       118
   macro avg       0.98      0.90      0.93       118
weighted avg       0.97      0.97      0.96       118

Accuracy: 0.9661016949152542
Precision: 0.9674202603160982
Recall: 0.9661016949152542
F1 Score: 0.9641482637062743
Mean Absolute Error: 0.03389830508474576
```

f. Record both 10-folds cross validation accuracy and prediction accuracy on test set result for comparison, removed hyper-tunning since model performance has been excellent.

Tabulate the results

```
]: from tabulate import tabulate

1]: cv_accuracies = ['CV Accuracy',cvScore_rf,cvScore_knn,cvScore_xgb,cvScore
    Names = ['Classifier', 'Random Forest', 'KNN', 'XgBoost', 'SVM', 'Logisti
    test_accuracies = ['Accuracy on testset', acc_test_rf,acc_test_knn,acc_te

2]: table = zip(Names, cv_accuracies, test_accuracies)

3]: print(tabulate(table, headers= 'firstrow', floatfmt=".4f" ,tablefmt = 'fa
```

Classifier	CV Accuracy	Accuracy on testset
Random Forest	0.9706	0.9661
KNN	0.9889	0.9915
XgBoost	0.9927	1.0000
SVM	0.9852	0.9831
Logistic Regression	0.8097	0.7797