Model Optimization and Tuning Phase Template

Date	July 2024
Team ID	Team-739815
Project Title	Power Consumption Analysis For Households
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

During the model optimization and tuning phase for power consumption analysis in households, the focus is on refining the predictive accuracy and performance of the models developed using techniques like linear regression or more advanced methods such as decision trees, random forests, or neural networks.

Hyperparameter Tuning Documentation (8 Marks):

Model	Tuned Hyperparameters

<table>
<tr>
<td>Linear

Regression</td>
<td>LinearRegression: The linear regression model used for regression tasks. It's part of the sklearn.linear_model module in the scikit-learn library.
lr.fit: This method is used to train the linear regression model.
X_train: The feature set used for training the model.
y_train: The target values corresponding to the training feature set.
X: The input features of the dataset generated using make_regression.
y: The target values of the dataset generated using make_regression.
X_train: The subset of X used for training the model.
X_test: The subset of X used for testing the model.
y_train: The subset of y used for training the model.
y_test: The subset of y used for testing the model.
y_pred: The predicted values generated by the model for the test set.
mse: The mean squared error calculated between the actual and predicted values of the test set. This metric is used to evaluate the model's performance.
<pre>
from sklearn.linear_model import LinearRegression

lr=LinearRegression()

lr.fit(X_train,y_train)

▾ LinearRegression
LinearRegression()
</pre>
</td>
</tr>
</table>

<table>
<tr>
<td>Random Forest</td>
<td>
n_samples=1000: Creates a dataset with 1000 samples.
n_features=10: Each sample will have 10 features.
noise=0.1: Adds a small amount of noise to the data to make it more realistic.
random_state=42: Ensures reproducibility of the results by setting a seed for the random number generator.
n_estimators=100: The number of trees in the forest.
random_state=42: Ensures reproducibility by using the same seed.
Fits the random forest model to the training data.

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Generate a random regression problem
X, y = make_regression(n_samples=1000, n_features=10, noise=0.1, random_state=42)

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create the random forest regressor
regressor = RandomForestRegressor(n_estimators=100, random_state=42)

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

```
▾ RandomForestRegressor
RandomForestRegressor(random_state=42)
```
</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>
n_samples=1000: Creates a dataset with 1000 samples.
n_features=10: Each sample will have 10 features.
noise=0.1: Adds a small amount of noise to the data.
random_state=42: Ensures reproducibility of the results by setting a seed for the random number generator.
X_train, X_test: Feature sets for training and testing.
y_train, y_test: Target values for training and testing.
test_size=0.2: 20% of the data is reserved for testing.
random_state=42: Ensures reproducibility by using the same seed.
Fits the decision tree model to the training data.

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Generate a random regression problem
X, y = make_regression(n_samples=1000, n_features=10, noise=0.1, random_state=42)

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create the decision tree regressor
regressor = DecisionTreeRegressor(random_state=42)

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

```
▾ DecisionTreeRegressor
DecisionTreeRegressor(random_state=42)
```
</td>
</tr>
</table>

Final Model Selection Justification (2 Marks):

<table>
<tr><th>Final Model</th><th>Reasoning</th></tr>
<tr><td>Random Forest</td><td>
Random Forest model is chosen for its robustness in handling complex datasets and its ability to mitigate overfitting while providing high predictive accuracy.
<table>
<tr><th></th><th>Name</th><th>Accuracy</th><th>f1_score</th><th>Recall</th><th>Precision</th></tr>
<tr><td>0</td><td>Logistic Regression</td><td>67.90</td><td>64.68</td><td>59.16</td><td>71.35</td></tr>
<tr><td>1</td><td>Decision Tree Classifier</td><td>73.88</td><td>66.60</td><td>52.41</td><td>91.32</td></tr>
<tr><td>2</td><td>Random Forest</td><td>74.68</td><td>66.70</td><td>51.03</td><td>96.24</td></tr>
<tr><td>3</td><td>K-Nearest Nieghbors</td><td>74.56</td><td>71.57</td><td>64.44</td><td>80.48</td></tr>
<tr><td>4</td><td>Xgboost</td><td>74.18</td><td>68.61</td><td>56.78</td><td>86.67</td></tr>
<tr><td>5</td><td>Ridge Classifier</td><td>68.39</td><td>63.91</td><td>56.32</td><td>73.87</td></tr>
</table>
Above all the models Random Forest model have the highest accuracy among all the models.
</td></tr>
</table>