Model Optimization and Tuning Phase Template

Date	July 2024
Team ID	740117
Project Title	Smart Home Temperature prediction using Machine Learning
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

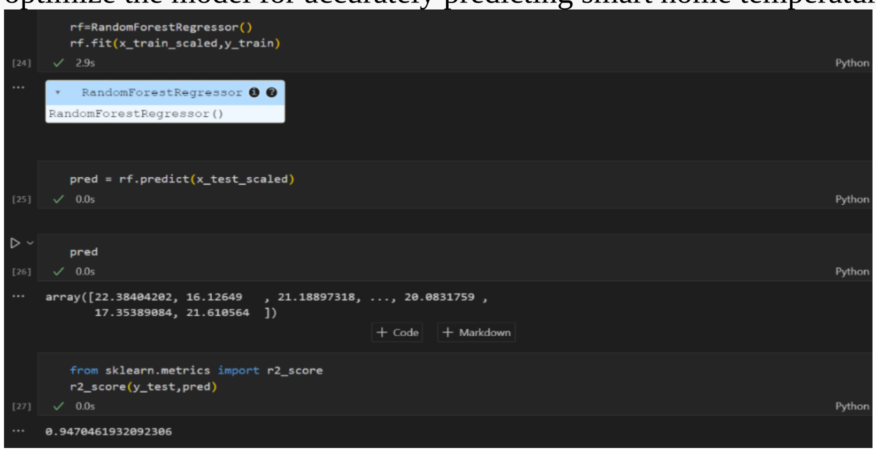
The Model Optimization and Tuning Phase involves refining neural network models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (8 Marks):

<table>
<tr><th>Model</th><th>Tuned Hyperparameters</th></tr>
<tr><td>Random Forest</td><td>

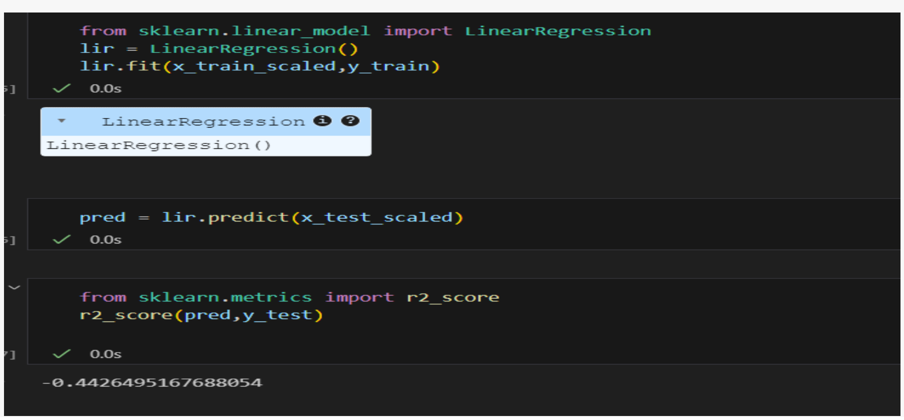
#importing RandomForestRegressor
from sklearn.ensemble import RandomForestRegressor

The parameter grid (param_grid) for hyperparameter tuning specifies different values for the number of trees (n_estimators), splitting criterion (criterion), maximum depth of trees (max_depth), and maximum features considered for splitting (max_features). The tuning process aims to optimize the model for accurately predicting smart home temperatures.



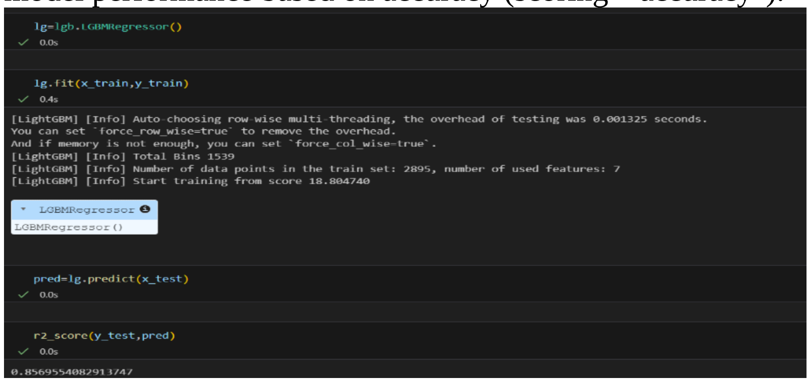
</td></tr>
<tr><td>Linear
Regression</td><td>

#importing LinearRegression
from sklearn.linear_model LinearRegression
The parameter grid (param_grid) for hyperparameter tuning specifies different values for the number of trees (n_estimators), splitting criterion (criterion), maximum depth of trees (max_depth), and maximum features considered for splitting (max_features). The tuning process aims to optimize the model for accurately predicting smart home temperatures.

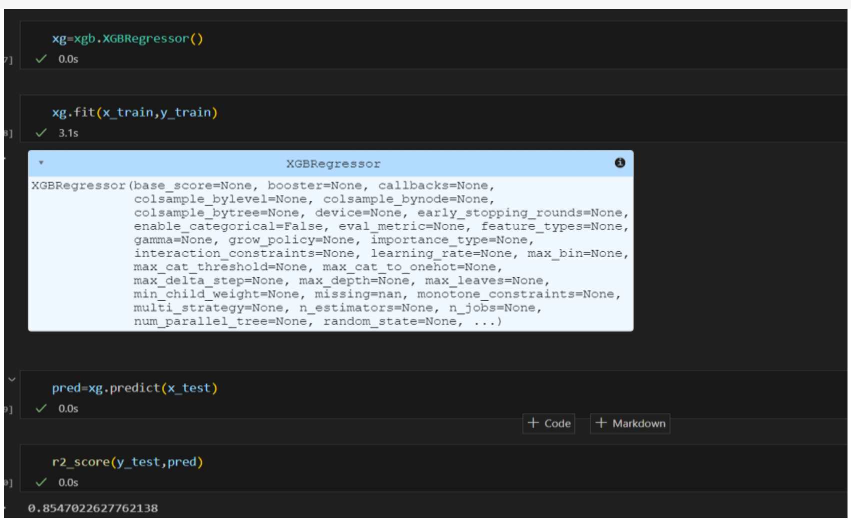


</td></tr>
</table>

<table>
<tr>
<td>LGB Regressor</td>
<td>
The parameter grid (params) for hyperparameter tuning specifies different values for min_child_weight, gamma, colsample_bytree, and max_depth. The tuning process aims to optimize the model for accurately predicting smart home temperatures. GridSearchCV is employed with 5-fold cross-validation (cv=5), refitting the best model (refit=True), and evaluating model performance based on accuracy (scoring="accuracy").

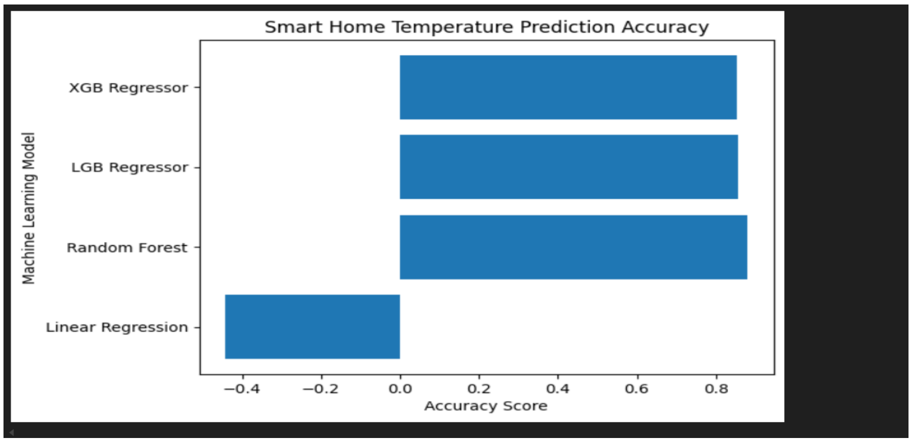


```
lg=lgb.LGBMRegressor()
✓ 0.0s

lg.fit(x_train,y_train)
✓ 0.4s

[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.001325 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 1539
[LightGBM] [Info] Number of data points in the train set: 2895, number of used features: 7
[LightGBM] [Info] Start training from score 18.804740

LGBMRegressor
LGBMRegressor()

pred=lg.predict(x_test)
✓ 0.0s

r2_score(y_test,pred)
✓ 0.0s

0.8569554082913747
```
</td>
</tr>
<tr>
<td>XGB Regressor</td>
<td>
The parameter grid (param_grid) for hyperparameter tuning specifies different values for the number of trees (n_estimators), splitting criterion (criterion), maximum depth of trees (max_depth), and maximum features considered for splitting (max_features). The tuning process aims to optimize the model for accurately predicting smart home temperatures.



```
xg=xgb.XGBRegressor()
✓ 0.0s

xg.fit(x_train,y_train)
✓ 3.1s

XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)

pred=xg.predict(x_test)
✓ 0.0s

+ Code   + Markdown

r2_score(y_test,pred)
✓ 0.0s

0.8547022627762138
```
</td>
</tr>
</table>

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Random Forest	Random Forest model is chosen for its robustness in handling complex datasets and its ability to mitigate overfitting while providing high predictive accuracy.  Above all the models Random Forest model have the highest accuracy among all the models.