

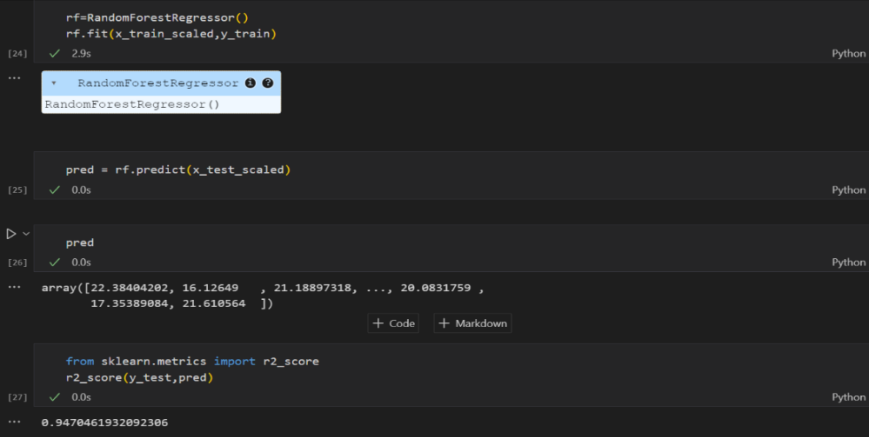
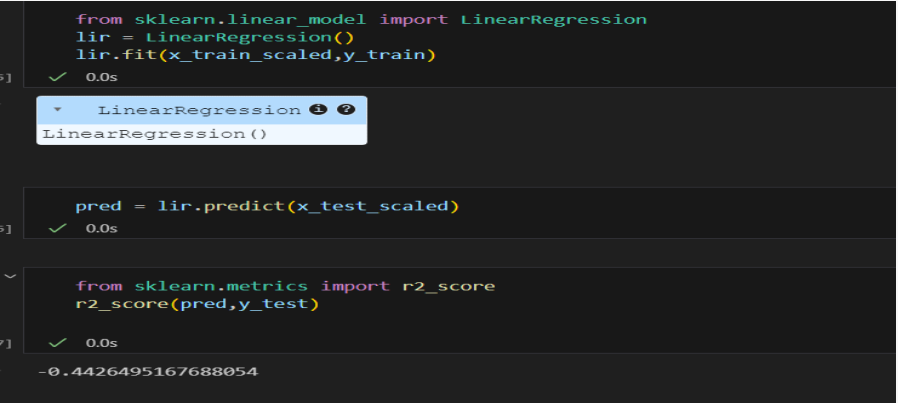
Model Optimization and Tuning Phase Template

Date	July 2024
Team ID	739670
Project Title	Smart Home Temperature prediction using Machine Learning
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

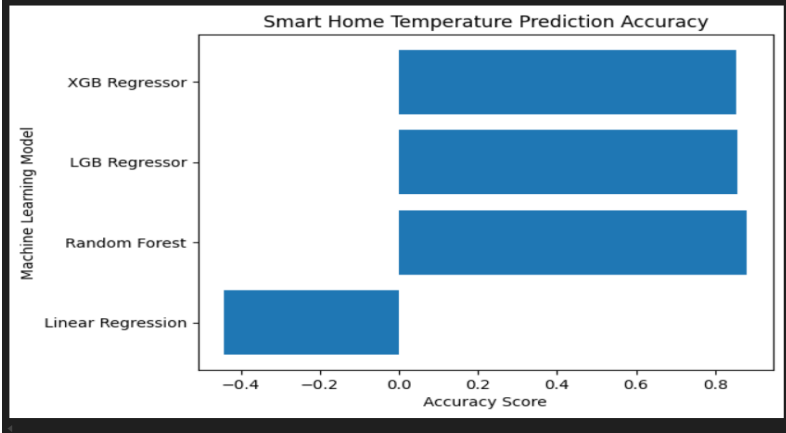
The Model Optimization and Tuning Phase involves refining neural network models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (8 Marks):

Model	Tuned Hyperparameters
Random Forest	<p>#importing RandomForestRegressor from sklearn.ensemble import RandomForestRegressor</p> <p>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.</p>  <p>The screenshot shows a Jupyter Notebook with the following code and output:</p> <pre> rfr=RandomForestRegressor() rfr.fit(x_train_scaled,y_train) [24] ✓ 2.9s Python + RandomForestRegressor ⓘ ⓘ RandomForestRegressor() pred = rfr.predict(x_test_scaled) [25] ✓ 0.0s Python pred [26] ✓ 0.0s Python array([[22.38404202, 16.12649 , 21.18897318, ..., 20.0831759 , 17.35389084, 21.610564]]) + Code + Markdown from sklearn.metrics import r2_score r2_score(y_test,pred) [27] ✓ 0.0s Python 0.9470461932092306 </pre>
Linear Regression	<p>#importing LinearRegression from sklearn.linear_model import LinearRegression</p> <p>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.</p>  <p>The screenshot shows a Jupyter Notebook with the following code and output:</p> <pre> from sklearn.linear_model import LinearRegression lir = LinearRegression() lir.fit(x_train_scaled,y_train) [1] ✓ 0.0s + LinearRegression ⓘ ⓘ LinearRegression() pred = lir.predict(x_test_scaled) [6] ✓ 0.0s from sklearn.metrics import r2_score r2_score(pred,y_test) [7] ✓ 0.0s -0.4426495167688054 </pre>

<p>LGB Regressor</p>	<p>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").</p> <pre> lgb=lgb.LGBMRegressor() ✓ 0.0s lgb.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() pred=lgb.predict(x_test) ✓ 0.0s r2_score(y_test,pred) ✓ 0.0s 0.856954082911747 </pre>
<p>XGB Regressor</p>	<p>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.</p> <pre> 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_bytrees=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_bin=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 r2_score(y_test,pred) ✓ 0.0s 0.8547022627762138 </pre>

Final Model Selection Justification (2 Marks):

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