Final Report

Team members: Qiuyi Zhang, Ninglin Huang, Yu Wang

1.Introduction

In this project, we worked on a modified dataset which describes the task of controlling a robot. The dataset includes 40 continuous features that characterize the robot's status, and the continuous target value describes the appropriate action corresponding to the feature. The objective of this project is to create a machine learning model that can correctly predict the target value. We already performed data exploration and evaluated different models in milestone 1 and performed data processing and feature engineering in milestone 2. Now, we will combine the previous work and select a proper model to tune its hyperparameters to build our own machine learning model. The final report includes the main steps we took.

2. Data Preprocessing and Feature Engineering

First, we draw the boxplot of all features, and found that acc_rate, track and climb_delta have obvious outliers. To reduce the noise before train the model, we drop the outliers of these three features. Then we removed duplicated rows and missing values. By removing these problematic data points, it can help improve the quality of the training data and increase the accuracy of the trained model.

During the previous work, we already found that the number of our features are too many and lots of features don't have high correlations with target. Instead of performing the feature selection by humans, we chose to use the SelectKBest() and VarianceThreshold() functions. The SelectKBest() will maintain the k(which is set as 30) features that most correlated with the target, and can help to reduce the dimensionality of the data and only retain the most informative features. VarianceThreshold() will discard the features which variances are lower than the threshold(which is set as 0.00001), and these features often don't contain much information and can cause noise in the training data. The choice of K and threshold was determined based on our multiple experiments and comparison of model performance. When these two parameters were set to the selected values, the resulting model performed better on the validation set and was able to retain the appropriate number and useful features. In addition, we also compared the correlated

feature pairs and discard time2 and time3, for they are highly correlated with time1. By doing this, it can reduce the number of features and prevent multicollinearity issues.

After feature selection, the maintained features are as follows:

['acc_rate', 'track', 'm', 'n', 'current_pitch', 'current_roll', 'absoluate_roll', 'climb_delta', 'roll_rate_delta', 'climb_delta_diff', 'time1', 'time4', 'time5', 'time6', 'time7', 'time8', 'time9', 'time10', 'time11', 'time12', 'time13', 'time14', 'omega', 'set']

We performed Standardization on train data and use the StandardScaler() to transform the test data, so all features contribute equally to the model. Then we performed PCA on training data which reducing the dimensionality of the training data to 11, and used the same PCA object to reduce the dimensionality of the test data. In this way, we reduced the number of features and retaining most of the information, which could result in better model performance and faster computation.

3. Choosing Model/Machine Learning Algorithm

By considering the evaluation of some baseline models and comparing their performance, we chose the Gaussian Process Regressor as our model.

We also evaluated KNN, random forest and LightGBM, but their obtained RMSE on the validation data are not as good as Gaussian Process Regressor. The result indicating that Gaussian Process Regressor had the best predictive performance on our dataset.

The following are the obtained RMSE of different models:

model	Validation RMSE
Gaussian Process Regressor	0.15360991057575307
KNN	0.18562716912058447
Random Forest Regressor	0.1765060438181095
LightGBM	0.1589557150214112

4. Hyperparameters Optimization

We performed the optimization of hyperparameters using Grid Search.

The Grid Search will generate a grid of all possible combinations of hyperparameters. For each combination of hyperparameters, it trained the model using k-fold cross-validation, and evaluated the performance by comparing the neg_mean_squared_error of models with different

hyperparameters. After exploring all search space, the method will find the best hyperparameters of Gaussian Process Regressor for our practice.

After the hyperparameters optimization process, it found the best hyperparameters for our model are as follows:

Best hyperparameters: {'alpha': 0.1, 'kernel': RBF(length_scale=10), 'random_state': [86]}

5. Conclusion

We split the training data as training data and validation data using train_test_split(). Then we used the best hyperparameters to train the model on training data, and used the validation data to get the performance of our model. We got the RMSE as 0.15360991057575307 on the validation data. Finally, we used the trained model to predict the target on test data and submit the result to classroom Kaggle competition. On the public dataset, we got a score of. 0.16348, which beat the baseline 2. This suggests that the model we trained is effective.

The code is saved in the notebook on Kaggle, which title is ‘Milestone3_QiuyiZhang’. You can check the code on GitHub, too.