Tutorial 3: Data Pre-processing and Evaluation

Universidad Carlos III de Madrid
Grado en Ingeniería Robótica
Machine Learning

Made by Ignacio González Meléndez (NIA 100522976)
& Lucas Beltrán Rúa (NIA 100523114)

Exercise 1: Features generation

1. If we execute the code from the last tutorial in one of the new maps the taxi won´t be able to complete its job. This is because our model was prepared specifically for map1, and the agent learnt how to act in map1, so he will go to passenger's location in map1 and try to pick him up, which is not possible.

2. Selected features:
   - Passenger_in_taxi: Binary variable (0 or 1) indicating whether the passenger has already been picked up. It replaces the need to interpret passenger indices.
   - Dist_to_passenger: Manhattan distance between taxi and passenger. Captures how far the taxi is from the passenger, it is independent of map layout. [abs(taxi_row - pass_row) + abs(taxi_col - pass_col)].
   - Dist_to_destination: Manhattan distance between taxi and destination. Indicates progress once the passenger is in the taxi. Also, as same as dist_to_passenger, it is independent for the map. [abs(taxi_row - dest_row) + abs(taxi_col - dest_col)].
   - Rel_passenger_row & rel_passenger_col: Difference in row and column between the taxi and the passenger. Help the model learn where to move. For example, if rel_passenger_row = -2, passenger is 2 cells above taxi.
   - Rel_dest_row & rel_dest_col: Difference in row and column between the taxi and the destination. Helps the model learn directional navigation toward the drop-off point and remains consistent across maps.

3. For the collecting process we divided the maps into 8 for train and 2 for test. Then using, using keyboard from pynput library, we played ten episodes per map and collected the original features and the additional features in each step. After completing all episodes in every map, we saved the original features in *dataset_original.csv*

and the additional ones in *dataset_additional.csv*. Resulting in around 1200 total steps.

4. Firstly, we selected Mutual Information, which measures the dependency between each feature and the reward. As can be seen in figure 1, the features with the most mi_score are *dist_to_destination, rel_dest_row* and *rel_dest_col*, which are the ones kept in *dataset_mi.csv*.
   The other feature selection method selected is Recursive Feature Elimination, which eliminates the least important features. Keeping with *dist_to_destination, rel_passenger_row, rel_dest_col* and saving them into *dataset_rfe.csv*. With all this, we help reduce redundancy and highlight the most informative features for the model.

FEATURES	MI_SCORE
dist_to_destination	0,291129
rel_dest_row	0,272770
rel_dest_col	0,263957
rel_passenger_col	0,193161
rel_passenger_row	0,141199
dist_to_passenger	0,122843
passenger_in_taxi	0,121729

Figure 1: MI score table

Exercise 2: Decision Tree with Cross Validation

1. In this part, we trained a Decision Tree model with some standard parameters (*"criterion": "entropy", "max_depth": 8, "min_samples_split": 10,"random_state": 42*) using 5-fold cross-validation on four datasets: the original, the one with additional features, and the ones created after feature selection (MI and RFE). As we can see in figure 2, the total correct predictions are 2813 and the incorrect ones are 1667. We can see that the best dataset predicted is additional with an accuracy of 0.8009. Also, we can see that the mean accuracy is over 60%, which means that we should improve the accuracy to get better predictions.

FEATURE	CORRECT	INCORRECT	ACCURACY	PRECISION	RECALL	F1
mi	685	435	0,6116	0,6892	0,7042	0,6943
additional	897	223	0,8009	0,8640	0,8421	0,8494
original	577	543	0,5161	0,5824	0,5632	0,5631
rfe	654	466	0,5839	0,6557	0,6527	0,6503
TOTAL	**2813**	**1667**				

Figure 2: Cross-Validation for each dataset table

2. For this question, we used the function *SearchGridCV,* also used in Tutorial 2, to find the best three models for each dataset. After this, we got the best parameters for each dataset and the best obtained accuracy:
    - Original dataset:
      Best CV Accuracy: 0.6339
      Best Parameters: {'class_weight': None, 'criterion': 'entropy', 'max_depth': 10, 'min_samples_split': 2, 'splitter': 'best'}
    - Additional dataset:
      Best CV Accuracy: 0.8759
      Best Parameters: {'class_weight': None, 'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 2, 'splitter': 'best'}
    - MI dataset:
      Best CV Accuracy: 0.5616
      Best Parameters: {'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 10, 'splitter': 'best'}
    - RFE dataset:

      Best CV Accuracy: 0.6304

      Best Parameters: {'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': None, 'min_samples_split': 2, 'splitter': 'best'}

   With this and looking at figure 3, we can conclude that the best models are the ones that used the additional dataset to train. In Figure 4, we can see the parameters of the best three models among all ones.

DATASET	MEAN_TEST_SCORE
original	0,579289
original	0,578453
original	0,578453
additional	0,820474
additional	0,817858
additional	0,814376
mi	0,524739
mi	0,52128
mi	0,520418
rfe	0,582631
rfe	0,58262
rfe	0,569731

Figure 3: Mean_test_score per dataset

DATASET	PARAMS	MEAN_TEST_SCORE
Additional1	{'class_weight': None, 'criterion': 'entropy', 'max_depth': None, 'min_samples_split': 2, 'splitter': 'best'}	0,875893
Additional2	{'class_weight': None, 'criterion': 'gini', 'max_depth': None, 'min_samples_split': 2, 'splitter': 'best'}	0, 874107
Additional3	{'class_weight': None, 'criterion': 'gini', 'max_depth': None, 'min_samples_split': 5, 'splitter': 'best'}	0, 868750

Figure 4: Top 3 models with cross-validation

3. Finally, we retrained these three models from figure 4, but in this case without using cross-validation and predicted the test data again. The results differ from the ones obtained in question 2 (see figure 4) because now we are training our model with 100% of training data. Meanwhile in question 2, the model was trained with less data per fold due to the cross-validation process. This slight difference can be seen comparing figure 4 with figure 5.

MODEL	CORRECT	ACCURACY(TEST)	PRECISION(TEST)	RECALL(TEST)	F1(TEST)
additional1	247	0,8821	0,8975	0,9057	0,8996
additional2	247	0,8821	0,8983	0,9047	0,8999
additional3	248	0,8857	0,9013	0,9079	0,9028
Total general	**742**				

Figure 5: Top 3 models test data prediction

Exercise 3: Deploy the agent

1. Yes, these models can reach the goal in some maps. This is because if an erroneous action is performed the agent won´t be able to complete its job as one action will affect the whole prediction, producing that the taxi will not complete his job. The fastest model should be number 3, which is the one with the most accuracy in figure 5, and in this simulation, it was not the fastest completing the task because model 2 managed to complete each successful map in 12 steps (Figure 6). Each model was able to complete 6 out of 8 maps. For every map, the taxi has a maximum of 100 steps.

Model	Train Successes	Train Total Maps	Train Success Rate	Avg Steps (Success)
Model_1_additional	6	8	0,75	14
Model_2_additional	6	8	0,75	12
Model_3_additional	6	8	0,75	13,5

Figure 6: Train maps agent deploys

2. No, the features do not make the problem general enough to solve the new configuration. This may be because our additional features are not general enough to complete new map configurations. Also, our models have learned the geometric distribution of the environment and not how to complete a general set of tasks to pick up the passenger and drop him off at the desired location. All this gives us a result of 0% rate of success in test maps, having a maximum of 100 steps for each map.

Conclusion

To sum up, our additional features were not enough to accomplish the taxi's job for a test map. To correct this, we could look for new general additional features which would train the agent to know exactly what to do in new situations, or we could try to teach the model how to act in unseen situations to get to the desired locations.