SF EXPRESS Automated Robotic Sorting System Based on Machine Learning

Xinyu Li

School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China, 100091
**Corresponding author. Email: 19281282@bjtu.edu.cn*

ABSTRACT
With the rapid development of Internet technology and the gradual expansion of logistics coverage, online shopping has become the preferred way of shopping. Facing the increasing volume of parcel express delivery, logistics companies need to efficiently complete the sorting of parcels. This paper aims to propose an intelligent express robot sorting system, which makes it possible to complete the sorting of express delivery through the program-controlled robot to efficiently and correctly complete the sorting of parcels in transit. The program adopts the method and principle of machine learning, performs data cleaning and feature extraction on part of the data before learning, and then it can perform high-accuracy work, and the accuracy rate can reach 97%.

Keywords: *machine learning, ensemble learning, feature filtering, data cleaning, parameter comparison.*

1. INTRODUCTION

In the Internet era, more and more express delivery companies have emerged, and the competition among express delivery industries has become increasingly fierce. After the service quality of major enterprises has reached the same level, the competition in the express delivery industry has also returned to the most primitive timeliness competition. Only faster timeliness can become the leader in the express delivery industry.

In the current industry, many companies are still conducting manual sorting, and the slow speed of manual sorting is the most important factor affecting the timeliness of express delivery. With the rapid development of artificial intelligence, automated sorting robots have greater demand in express delivery companies. However, it takes a lot of time, money, and high cost to develop a system for controlling sorting robots, and finally release it through the steps of feasibility analysis, demand analysis, system development, and testing. Therefore, if a proven open-source system is provided, it can help express companies quickly complete the transition from manual sorting to robotic sorting. By collecting data in express companies, using machine learning-related methods, data processing, and integrated learning, the development of corresponding systems can be completed quickly and efficiently, and the success rate of machine sorting can be guaranteed to a certain extent, helping companies in the sorting process. The speed has been greatly improved, and the error rate has been greatly reduced, thereby improving production efficiency and labor costs[1]. So this paper presents an intelligent express robot sorting system, which can complete the express sorting through a program-controlled robot, so as to complete the parcel sorting in transportation efficiently and correctly. In the future, it will be a general trend that machine sorting will replace manual sorting, and the system will also help some companies with insufficient funds to develop their own automatic sorting system.

2. METHODOLOGY

2.1. Datasets

24 ultrasonic sensors arranged in a circle were used to collect data as the SF sorting robot traveled clockwise along the wall inside the express processing center. The number of data pieces finally collected is about 5,000 pieces, including 24 attribute columns of sensor data and 1 label. Among them, US0 represents the ultrasonic reading of the first sensor, which is the ultrasonic sensor on the front of the robot, and the reference angle is 180°. US1 is the ultrasonic reading of the second sensor, and the reference angle is -165°.

The reference angles of subsequent US2-US23 are respectively increased by 15° on the basis of US1, which is-150°, -135°, -120°, -105°... US12 represents the

ultrasonic reading of the ultrasonic sensor located on the back of the robot, the reference angle is 0°. The label column includes four possible values: - Slight-Right-Turn, - Sharp-Right-Turn, - Move-Forward, - Slight-Left-Turn. The first five rows of the dataset are shown in table 1 below.

Table1. Original dataset

US0	US1	US2	US3	US4	US5	US6-20	US21	US22	US23	Class
0.438	0.498	3.625	3.645	5.000	2.918	...	0.444	0.440	0.429	Slight-Right-Turn
0.438	0.498	3.625	3.648	5.000	2.918	...	0.444	0.443	0.429	Slight-Right-Turn
0.438	0.498	3.625	3.629	5.000	2.918	...	0.444	0.446	0.429	Slight-Right-Turn
0.437	0.501	3.625	3.626	5.000	2.918	...	0.444	0.444	0.429	Slight-Right-Turn
0.438	0.498	3.626	3.629	5.000	2.918	...	0.444	0.441	0.429	Slight-Right-Turn

Simple feature engineering is carried out on the collected original data set, and four sensor readings called "simplified distance" and corresponding class labels are added (the last column, the labels are the same as in the first dataset). These simplified distances are called "SD_front", "SD_right", "SD_left" and " SD_back". They consist of the smallest sensor readings among the sensors located within a 60-degree arc in the front, left, right, and rear of the robot, respectively. The first five columns of partial data in the resulting dataset are shown below.

Table 2. Dataset after processing

US0	US1	US2	US3	....	US23	SD_front	SD_right	SD_left	SD_back	Class
0.438	0.498	3.625	3.645	....	0.429	1.687	0.445	2.332	0.429	Slight-Right-Turn
0.438	0.498	3.625	3.648	....	0.429	1.687	0.449	2.332	0.429	Slight-Right-Turn
0.438	0.498	3.625	3.629	....	0.429	1.687	0.449	2.334	0.429	Slight-Right-Turn
0.437	0.501	3.625	3.626	....	0.429	1.687	0.449	2.334	0.429	Slight-Right-Turn
0.438	0.498	3.626	3.629	....	0.429	1.687	0.449	2.334	0.429	Slight-Right-Turn

2.2. Data cleaning

Checking whether the data has missing values or repeated samples. If there are no missing values, the next step of analysis can proceed. Statistically, there are 0 missing sample data in the data, confirming that these data can continue to be used[2]. Convert each category of Class to a number, where Move-Forward is converted to 1, Sharp-Right-Turn is converted to 2, Slight-Right-Turn is converted to 3, and Slight-Left-Turn is converted to 4. Finally, viewing the distribution of the characteristics of each column in the form of a table and a histogram.

Table 3. The distribution of features for each column in the dataset

	count	mean	std	min	25%	50%	75%	max
SD_front	5456.0	1.290307	0.626700	0.495	0.82600	1.0895	1.51950	5.000
SD_left	5456.0	0.681269	0.342594	0.340	0.49500	0.6120	0.75300	5.000
SD_right	5456.0	1.881819	0.562533	0.836	1.47200	1.7530	2.13900	5.000
SD_back	5456.0	1.273689	0.821750	0.367	0.78800	1.0665	1.40050	5.000
US0	5456.0	1.471617	0.802801	0.400	0.92100	1.3350	1.81400	5.000
US1	5456.0	2.327043	1.410146	0.437	1.36200	1.9045	2.68150	5.025

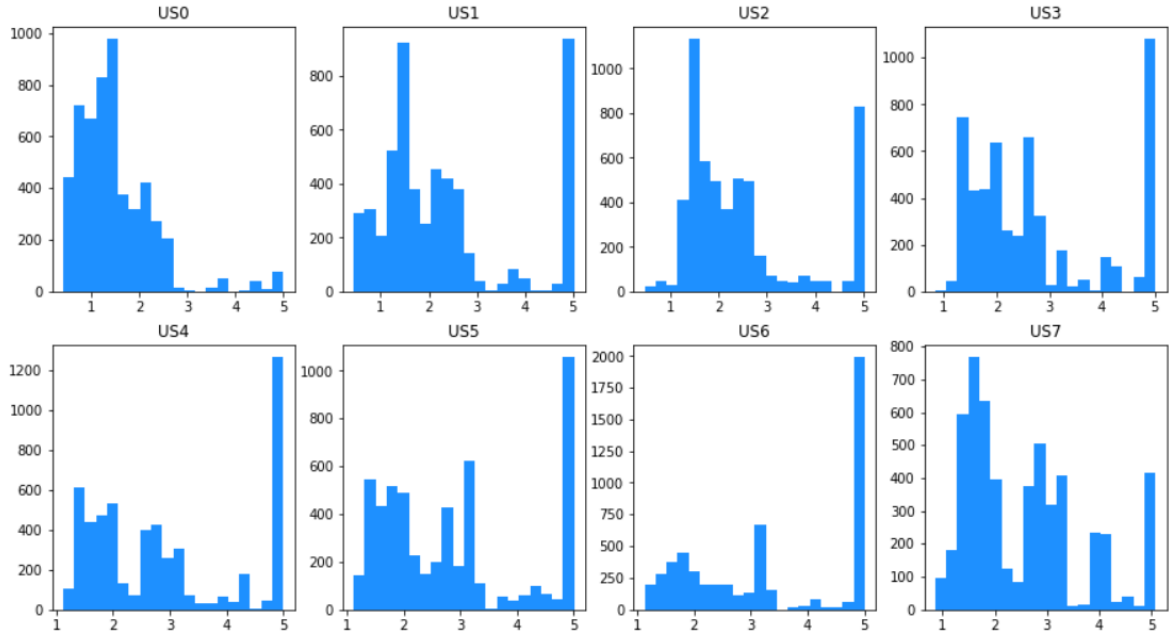

Figure1 The distribution of features for each column in the dataset

2.3. Feature Engineering

Firstly, the features and labels should be separated, and the data should be shuffled through the shuffle function and the training set and the test set should be divided (division ratio 8:2). The conclusion can be drawn from the observation data in section 2.3:

- The dimensions of the data are not very different.
- Data are free of duplications and deletions.
- Features are all continuous values.

2.3.1. Discretize continuous features

Therefore, the next attempt is to discretize continuous features. Consider 3 methods:

1)Equidistant discretization, grouping continuous features at fixed intervals;

2)Equal frequency discretization, grouping continuous features by a fixed number of samples;

3)Using algorithms or statistical indicators to discretize, such as chi-square values, decision tree, cluster analysis, etc.

Here, the general realization principles of the three methods are given first, and the influence of different discretization methods on the results will be compared in the analysis of the results.

Method 1: Equidistant Divergence-Set two functions. The first one records the boundary value when the data set is divided into several parts. The second one calls the first function to record the boundary value of the training set, applies it to the training set and test set for binning and returns the binned training set and test set[3].

Method 2: Equal frequency discretization-divide the data into several parts after sorting, training and test are the data sets, bins are the number of parts, and return the training set and test set after binning[3].

Method 3: Using decision tree to implement feature binning and build 3 functions.

def cut_bin(df, label, max_depth, percent):

```
#
Function: bin the training set
Parameter Description
input:
df: training set name, such as train;
label: label name, such as 'Class';
max_depth: maximum depth of decision tree;
percent: the ratio of the minimum number of leaf
nodes to the sample size
output:
df_bin: data set after binning;
dict_bin: dictionary storing binning parameters
#
```

def cut_test_bin(df, label, train_dict_bin):

```
#
Function: binning the test set
Parameter Description
input:
df: test set name, such as test;
label: label name, such as 'Class';
train_dict_bin: dictionary of storage parameters
generated by training set binning
output:
df_bin: data set after binning;
dict_bin: dictionary storing binning parameters
#
```

The function DT_cut applies the first two functions to the data set. The parameters are the training set name of the feature and the label. The test set name of the feature and the label, the maximum depth of the decision tree, and the ratio of the minimum number of leaf nodes to the sample size, and returns the divided dataset[4].

The main idea of the 3 functions is to use the discretized variables to train a decision tree of finite depth (2, 3 or 4) to predict the target. Then replace the original variable value with the probability returned by the decision tree. All observations within a single bin have the same probability, so replacing with probability is equivalent to grouping observations within the cutoff determined by the decision tree.

2.3.2. feature filtering

After feature discretization, basic PCA dimensionality reduction is used for feature filtering[5].

```
from sklearn.decomposition import PCA
pca = PCA(25)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)
```

At this point, the feature engineering work is over, and the processing of the dataset is completed.

3. MODEL CONSTRUCTION AND TRAINING

3.1. Basic model construction

The next step of the package will be predicted through the currently processed sensor data to achieve the purpose of sorting, so as to form structured data and model by the structured data.

For the data, the model will be built in the following order[6]:

- Select models for cross-validation and grid search
- Model Ensemble - Stacking and Bagging Methods
- Fine-tune the threshold
- Scorecard Conversion

Firstly, building a logistic regression model, and setting the hyperparameters to be tuned:

- Regularization coefficient C
- Regularization type penalty

Using the grid search parameter GridSearchCV to adjust the hyperparameters, and the final model result is LogisticRegression('C':1, 'penalty': 'l2').

The key code is

```
##Grid search, computing time
begin = time.time()
lr = LogisticRegression()
params = {'C':[0.1,1,10,100],'penalty':['l1','l2']}
#  grid search
gs_lr = GridSearchCV(estimator =
lr,param_grid = params,cv = 10,scoring =
make_scorer(accuracy_score))      #10 fold cross
gs_lr.fit(X_train_pca, y_train)      #  train
end = time.time()
#  check out the optimal parameters
print(gs_lr.best_params_)
#  check the indicators under the optimal
parameters
print(gs_lr.best_score_)
print(' The time consumed by grid search i:%.5f
s'% float(end - begin))
```

Next, building the random forest classification model RandomForestClassifier, and setting the hyperparameters that need to be adjusted:

- Maximum number of features max_features
- Decision tree depth max_depth
- decision tree trees n_estimators

Using the grid search parameter GridSearCV to adjust the hyperparameters, and the final model result is *RandomForestClassifier(max_depth=16, max_features=9, n_estimators=2048, n_jobs=-1, random_state=0)*

In addition, a random guessing function is designed as a baseline to further see whether the model has a great advantage over random guessing, and further comparisons are made by comparing the accuracy score and AUC values to judge whether the model has sufficient accuracy[7]. The specific values are shown in the table below.

Table 4. Comparison table of AUC and accuracy score of different models

	AUC score	accuracy score
Logistic	0.923988	0.749084
RandomForest	0.978049	0.899267
random prediction	0.473522	0.232601

By comparing the accuracy rate and AUC value, it can be seen that the RandomForest model has a better fit for the data set, and both RandomForest and Logistic have better accuracy than the ordinary random model. The accuracy of random forest is 0.899, which is 0.15 higher than the accuracy of the logistic method and 0.67 higher than that of the ordinary random method of 0.233. It can be concluded that using these two models can greatly improve the accuracy of prediction.

3.2. Integrated Learning

On the basis of the above, the method of ensemble learning is considered to further improve the predictive ability of the model[8]. Three ensemble learning methods, Adaboost, Bagging, and stacking are included as the selection target.

3.2.1. AdaBoost

AdaBoost is a typical representative of the boosting family of algorithms, which can upgrade a weak learner to a strong learner. Firstly, a base classifier is trained with the initial training set, and then the weights of all samples are re-adjusted to make the current base classifier. The wrongly classified samples receive special attention, and the weight of the current base classifier is set according to the prediction error of the current base classifier. Then, the weight-adjusted samples are retrained for the base classifier, and the above process is repeated step by step until the specified number of times is reached, and finally, the prediction results of multiple base classifiers are weighted and combined[9]. The general flow of its algorithm is as follows.

input: Training set D = {(x1, y1), (x2, y2), …, (xm,ym)};
Basic Learning Algorithms δ
number of training rounds T.
process:
1: $D_1(x) = 1/m.$
2: **for** *t = 1, 2, ..., T* **do**
3: $h_t = \delta(D, D_t);$
4: $\varepsilon_t = P_{x\sim D_t}(h_t(x) \neq f(x));$
5: **if** $\varepsilon_t > 0.5$ **then break**
6: $\alpha_t = \frac{1}{2}\ln(\frac{1-\varepsilon_t}{\varepsilon_t});$
7: $D_{t+1}(x) = \frac{D_t(x)}{Z_t} \times \begin{cases} exp(-\alpha_t), & if\ h_t(x) = f(x) \\ exp(\alpha_t), & if\ h_t(x) \neq f(x) \end{cases} = \frac{D_t(x)exp(-\alpha_t f(x)h_t(x))}{Z_t}$
8: **end for**
output: $H(x) = sign(\sum_{t=1}^{T}\alpha_t h_t(x))$

Initializing the sample weight distribution, and training the classifier h_t from the dataset D based on the distribution Dt;

Error in estimating ht;

Determining the weights of the classifier ht;

Updating the sample distribution, where Zt is the normalization factor to ensure that Dt+1 is a distribution.

Finally, the MSE performance on the training set and test set using the AdaBoost algorithm is 0.918 and 1.064, respectively. It can be found that there are still some overfitting situations.

3.2.2. Bagging

The Bagging algorithm process is rough as follows:

• Extracting the training set from the original sample set. Each round uses the Bootstrapping method to draw n training samples from the original sample set (in the training set, some samples may be drawn multiple times, and some samples may not be drawn at all)[10]. A total of k rounds of extraction are performed to obtain k training sets. (The k training sets are independent of each other)

• Each time a training set is used to obtain a model, and k training sets are used to obtain k models in total[10]. (Note: There is no specific classification algorithm or regression method here, we can use different classification or regression methods according to specific problems, such as decision tree, perceptron, etc.)

• For classification problems: the k models obtained in the previous step are voted to obtain the classification results - the category or one of the categories with the most votes is the final model output; for regression problems, the mean of the above models is calculated as the final result. (Equal importance for all models)

The MSE performance of the bagging algorithm on the training set and test set is 0.637 and 0.648 respectively.

3.2.3. Stacking

Stacking is a layered model integration framework. Taking two layers as an example, the first layer consists of multiple base learners whose input is the original training set, and the model of the second layer uses the output of the first layer of base learners as features to add to the training set for retraining, so as to get the full stacking model.

The stacking algorithm flow is as follows:

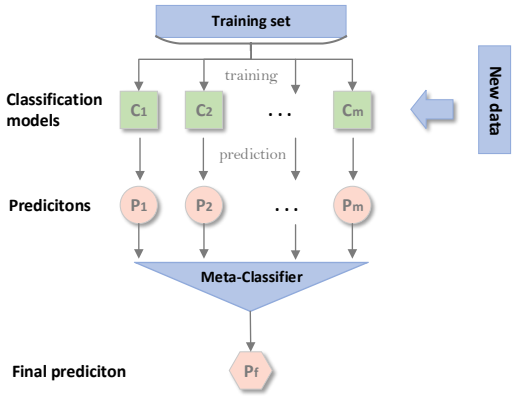

Figure 2 Flow chart of stacking algorithm

The grid-searched logistic and random forest models are selected as the base learner, and the simplest logistic model is used as the secondary learner. The MSE

performance of the final Stacking algorithm on the training and testing sets is 0.832 and 1.032, respectively.

Table 5. Table of Different Integration Methods

MSE	training set	test set
AdaBoost	0.918	1.064
Bagging	0.637	0.648
Stacking	0.832	1.032

To sum up, it can be seen that Bagging has the best effect, and the results obtained by the other two integration methods are close, but the overfitting problem caused by stacking is larger.

4. RESULT

The optimal model finally obtained is through Bagging. Here, the ROC curve of the optimal model on the test set is drawn, and it is found that the effect is still very good. Therefore, the system can indeed be used by express companies to help improve the sorting speed, and the accuracy is also very objective, which can greatly reduce the demand for personnel.

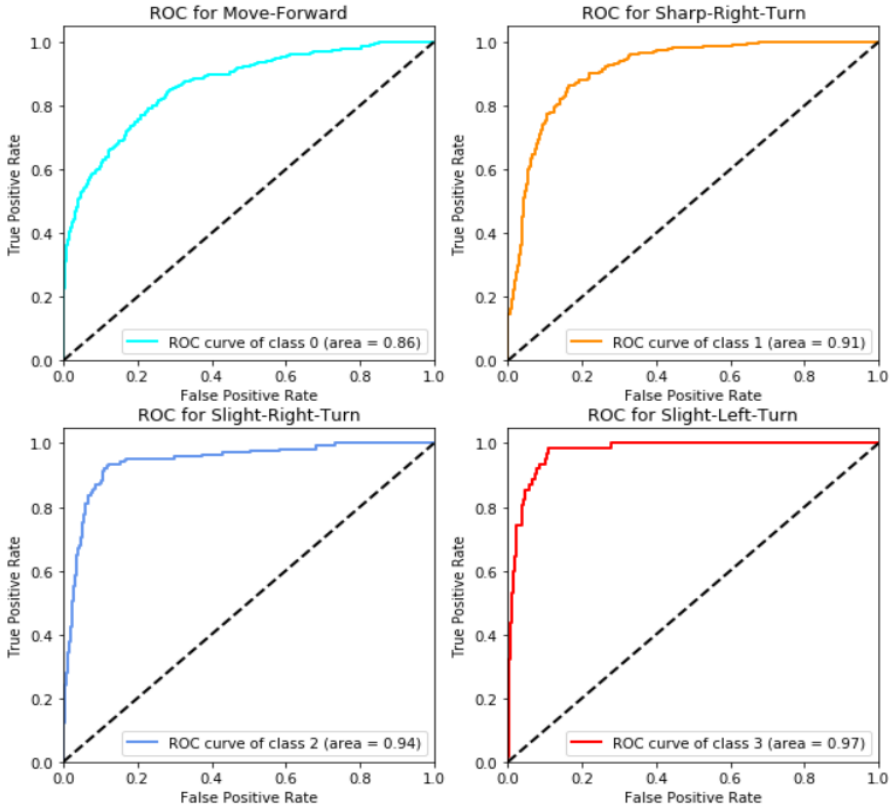

Figure 3 The ROC curve of the optimal model

5. CONCLUSION

Therefore, the automated robotic sorting system proposed in this paper can help enterprises improve sorting speed and reduce labor costs. At the same time, it is more convenient to develop the system according to the steps described in this article, and it can be completed quickly. Of course, the system has only been studied and tested on the more than 5,000 pieces of data we have collected. If it is to be applied to the industry on a large scale, further verification may be required based on more data. Whether the sorting accuracy can always maintain a good level after large-scale application, or even further improve it, is a question that can be further studied in the future.

REFERENCES

[1] Shalev-Shwartz, S. and S. Ben-David. (2014). Understanding Machine Learning. Cambridge University Press, Cambridge, UK.

[2] Witten, I. H., E. Frank, and M. A. Hall. (2011). Data Mining: Practical Machine Learning Tools and Techniques, 3rd edition. Elsevier, Burlington, MA. Wolpert, D. H. "The lack of a priori distinctions between learning algorithms." Neural Computation, 8(7) (1996) 1341-1390.

[3] Blum, A. and P. Langley. "Selection of relevant features and examples in machine learning." Artificial Intelligence, 97(1-2) (1997) 245-271.

[4] Murthy, S. K. (1998). "Automatic construction of decision trees from data: A multi-disciplinary survey." Data Mining and Knowledge Discovery 2(4): 345-389.

[5] Zhang, D. and Z.-H. Zhou. "$(2D)^2$ PCA: 2-directional 2-dimensional PCA for efficient face representation and recognition." Neurocomputing, 69(1-3) (2005) 224-231.

[6] Escalera, S., O. Pujol, and P. Radeva. "Error-correcting output codes library." Journal of Machine Learning Research, 11, 2010, pp. 661-664.

[7] Wellek, S. Testing Statistical Hypotheses of Equivalence and Noninferiority, 2nd edition. Chapman & Hall/CRC, Boca Raton, FL. 2010.

[8] Rokach, L. Pattern Classification Using Ensemble Methods. World Scientific, Singapore. 2010b.

[9] Schapire, R. E. and Y. Freund. Boosting: Foundations and Algorithms. MIT Press, Cambridge, MA. 2012.

[10] Breiman, L. "Bagging predictors". Machine Learning, 24(2) (1996a) 123-140.