NTHU

Project 2

On-line Shopping Intention Prediction

指導教授：邱銘傳　教授

Group3: 10803446 胡理嫚

Group3: 10803454 李佩怡

Group3: 10803436 朱文伃

1. Motivation and purpose

Owing to the lower level of sale volumes of the washing machine on the Internet last year, the manager of the marketing department in ABC household electric appliance company requires us to put forward solutions to elevate the level of sales volume. To achieve this, we analyze the buying behaviors of customers, predict the purchasing intention, find out the root causes of low sales rate and the opportunities of improvement, and apply the marketing methods to increase the purchasing intention by collecting and analyzing all the information about customers' activities on the related shopping websites.

2. Dataset description

In our project, the purchasing intention model is designed as a binary classification problem measuring the user's intention to finalize the transaction. The dataset consists of feature vectors belonging to 12,330 sessions [1]. The dataset was formed so that each session would belong to a different user in a 1-year period to avoid any tendency to a specific campaign, special day, user profile, or period. Of the 12,330 sessions in the dataset, 84.5% (10,422) were negative class samples that did not end with shopping, and the rest (1908) were positive class samples ending with shopping. The dataset used for model training contains 18 features, including Administrative, Administrative, Duration, Informational, Informational duration, Product Related, Product Related, Duration, Bounce rates, Exit Rates, Page Values, Special Day, Month, Operating systems, Browser, Region, Traffic type, Visitor type, Weekend, Revenue. A detailed introduction is shown in Table 1 and Table 2.

Table 1. Part 1 of dataset description

Feature name	Feature description	Min value	Max value	SD
Administrative	Number of pages visited by the visitor about account management	0	27	3.32
Administrative duration	Total amount of time (in seconds) spent by the visitor on account management related pages	0	3398	176.70
Informational	Number of pages visited by the visitor about Web site, communication and address information of the shopping site	0	24	1.26
Informational duration	Total amount of time (in seconds) spent by the visitor on information pages	0	2549	140.64
Product Related	Number of pages visited by visitor about product related pages	0	705	44.45

Product Related duration	Total amount of time (in seconds) spent by the visitor on product related pages	0	63973	1912.25
Bounce rate	Average bounce rate value of the pages visited by visitor	0	0.2	0.04
Exit rate	Average exit rate value of pages visited by visitor	0	0.2	0.05
Page value	Average page value of the pages visited by the visitor	0	361	18.55
Special day	Closeness of the site visiting time to a special day	0	1.0	0.19

Table 2. Part 2 of dataset description

Feature name	Feature description	Number of categorical values
Operating Systems	Operating system of the visitor	8
Browser	Browser of the visitor	13
Region	Geographic region from which the session has been started by the visitor	9
Traffic Type	Traffic source by which the visitor has arrived at the Web site (e.g., banner, SMS, direct)	20
Visitor Type	Visitor type as "New Visitor," "Returning Visitor," and "Other"	3
Weekend	Boolean value indicating whether the date of the visit is weekend	2
Month	Month value of the visit date	12
Revenue	Class label indicating whether the visit has been finalized with a transaction	2

3. Oversampling

When implementing classification algorithms, the structure of the data is of great significance. Specifically, the balance between the number of observations for each potential output heavily influences the prediction's performance. Trying to make predictions with regards to a minority class, we use SMOTE which is a technique based on nearest neighbors to make the number of observations balanced.

SMOTE is a technique based on nearest neighbors judged by Euclidean Distance between data points in feature space. It has four steps for implementations:

Step 1. For each minority instance, k number of nearest neighbors are found such that they also belong to the same class where k=SMOTE%/100.

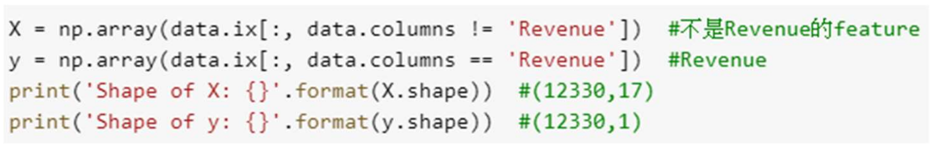
Step 2. The difference between the feature vector of the considered instance and the feature vectors of the k nearest neighbors are found. As a consequence, k number of difference vectors are obtained.

Step 3. The k difference vectors are each multiplied by a random number between 0 and 1 (excluding 0 and 1).

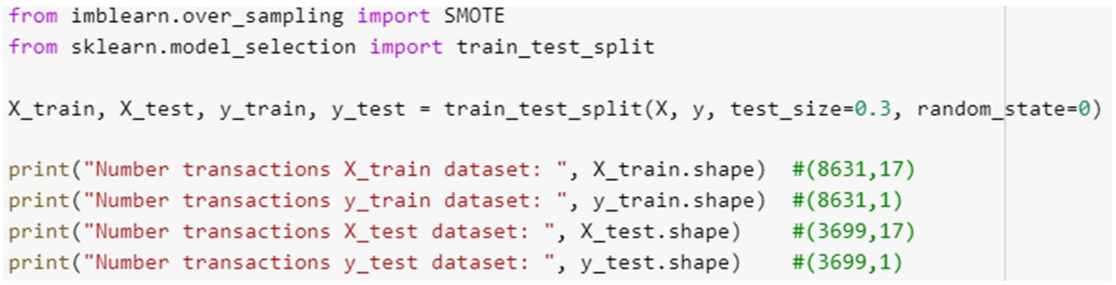
Step4. The difference vectors, after being multiplied by random numbers, are added to the feature vector of the considered instance (original minority instance) at each iteration. (See code in Fig. 1)

Since the dataset is created by selecting multiple instances of the minority class more than once, first oversampling the dataset and then dividing it into training and test sets may lead to biased results due to the possibility that the same minority class may be used both for training and testing. For this reason, in this project, 30% (3699) of the dataset consisting of 12330 samples is first left out for testing and the oversampling method is applied to the remaining 70% (8631) of the samples.

Figure 1. The code of oversampling-SMOTE.

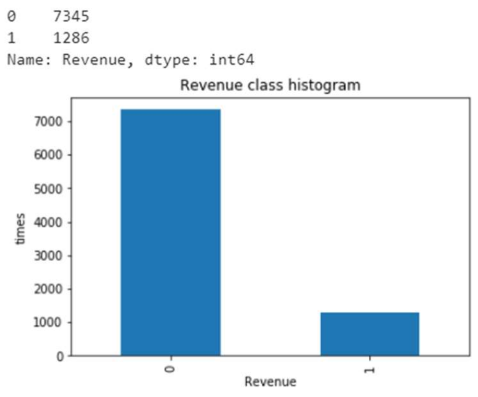


```
X = np.array(data.ix[:, data.columns != 'Revenue'])  #不是Revenue的feature
y = np.array(data.ix[:, data.columns == 'Revenue'])  #Revenue
print('Shape of X: {}'.format(X.shape))  #(12330,17)
print('Shape of y: {}'.format(y.shape))  #(12330,1)
```



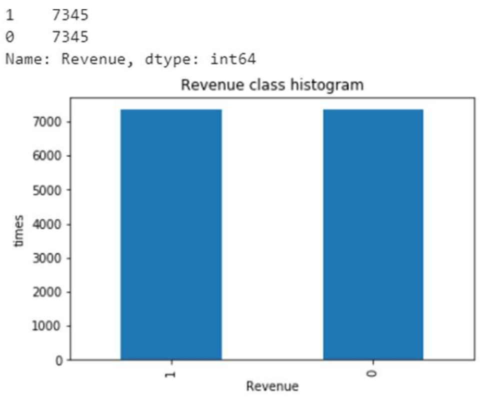
```
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

print("Number transactions X_train dataset: ", X_train.shape)  #(8631,17)
print("Number transactions y_train dataset: ", y_train.shape)  #(8631,1)
print("Number transactions X_test dataset: ", X_test.shape)    #(3699,17)
print("Number transactions y_test dataset: ", y_test.shape)    #(3699,1)
```

Before oversampling, the number of class 0 is 1286 and 7345 of class 1; after oversampling, the number of class 0 and class 1 are both 7345. (See Fig. 2 and Fig. 3)



```
0    7345
1    1286
Name: Revenue, dtype: int64
```

Figure 2. Before oversampling



```
1    7345
0    7345
Name: Revenue, dtype: int64
```

Figure 3. After oversampling

4. Feature Selection

The two biggest problems in machine learning are overfitting (in terms of fitting data that cannot be generalized outside the data set) and dimensional disasters (non-intuitive and sparse in high-dimensional data). By reducing the number of features in the model and trying to optimize model performance, feature selection helps avoid these two problems. In this way, feature selection provides an additional benefit: model interpretation. With fewer features, the output model becomes simpler and easier to interpret, and makes it easier to trust the predictions made by the model.

In this section, we use the feature selection method to filter out important features to further improve the classification performance of MLP, SVM, RNN, and CNN algorithms. Besides, because using fewer features to achieve better or similar classification results will also improve the scalability of the real-time online shopper behavior analysis system. Another method for feature selection is to use feature extraction (such as principal component analysis) for dimensionality reduction. However, in this case, the features in the reduced space will be a linear combination of 17 attributes, which requires tracking all features during the visit and updating the feature vector after the visitor takes a new operation. Therefore, it is considered appropriate to use feature selection instead of feature extraction within the scope of this study.

For feature ranking, we tend to apply filter-based feature selection and wrapper algorithms. The difference between the two methods is that the wrapper algorithm requires a learning algorithm to obtain a simplified feature set for a specific classifier. Here, we use correlation, minimum redundancy maximum relevance (mRMR) filters and a wrapper method—random forest (RF) for feature selection.

4.1 Correlation

This method filters and obtains only a subset of the relevant features. The filtering here is done using a correlation matrix, which is usually done using Pearson Correlation. Here, we first draw a Pearson Correlation heat map (shown in Fig. 4) and look at the correlation between the independent variable and the output variable MEDV. We only select features that have a correlation with the output variable greater than 0.09 (taken as an absolute value). In correlation matrix, the value of the correlation coefficient is between -1 and 1: a value close to 0 indicates weak correlation (exact 0 means no correlation), a value close to 1 means strong positive correlation, and a value close to -1 means strong negative correlation.

4.2 mRMR

Unlike Correlation, a univariate filter method, mRMR is a multivariate filter method, which is heuristic and can be used along with other methods (such as the wrapper method). mRMR can use mutual information, correlation or distance/similarity scores to select features. Its goal is to punish the relevance of features by their redundancy in the presence of other selected features. We use the mRMR software package provided by Peng et al. [1] to implement feature selection and select 9 important features.

4.3 Random Forest

In data science work, RF is a commonly used feature selection method. The idea of feature importance evaluation is to see how much each feature contributes to each tree in a random forest, then take an average, and finally, the ratio of the contributions between features is compared. Contribution can usually be measured using the Gini index or OOB error rate as an evaluation indicator. In this project, we call "RandomForestClassifier ()" in the scikit-learn random forest class library to filter and sort the features. The results are shown in Fig. 5. The most important feature shown is the eighth feature, "PageValues".

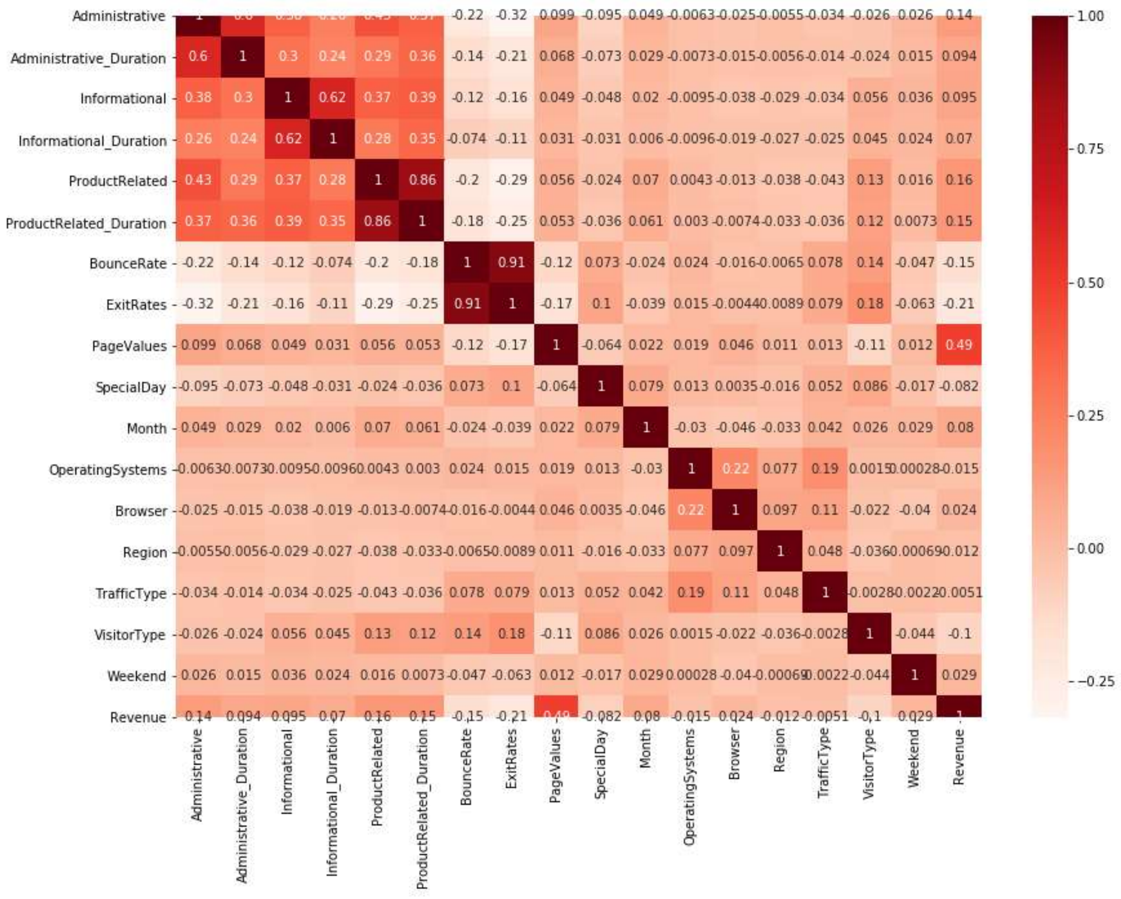


Figure 4. Pearson correlation heat map

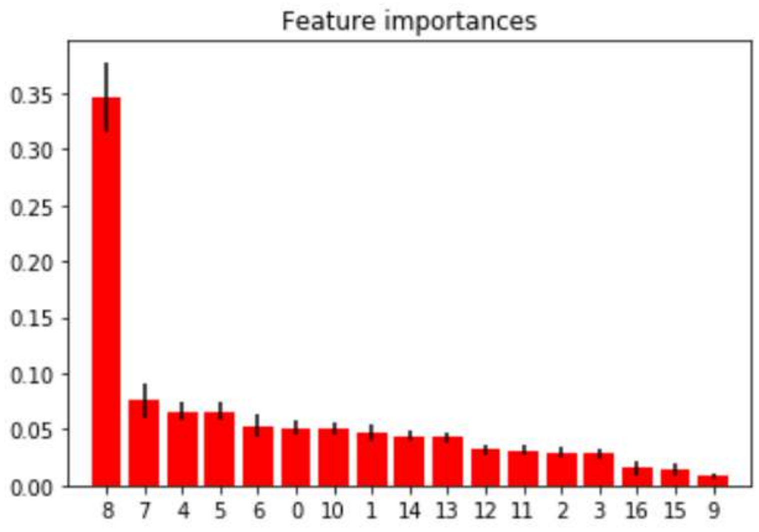


Figure 5. Feature selection ranking results from RF

According to the above three feature selection methods, Table 3 summaries the ranking of the features selected from different methods.

Table 3. Feature ranking

Rank	Filter-based feature selection		Wrapper method
-ing	Correlation	mRMR	RF
1	PageValues	ProductRelated_Duration	PageValues
2	ExitRates	ExitRates	ExitRates
3	ProductRelated	BounceRates	ProductRelated
4	ProductRelated_Duration	SpecialDay	ProductRelated_Duration
5	BounceRate	VisitorType	BounceRate
6	Administrative	Weekend	Administrative
7	VisitorType	PageValues	Month
8	Informational	OperatingSystems	Administrative_Duration
9	Administrative_Duration	Informational	TrafficType
10	SpecialDay	Month	Region
11	Month	Region	Browser
12	Informational_Duration	Browser	OperatingSystems
13	Weekend	Administrative	Informational
14	Browser	TrafficType	Informational_Duration
15	OperatingSystems	Informational_Duration	Weekend
16	Region	ProductRelated	VisitorType
17	TrafficType	Administrative_Duration	SpecialDay

5. Prediction of online shoppers' purchasing intention

5.1 Multilayer perceptron (MLP)

5.1.1 MLP definition

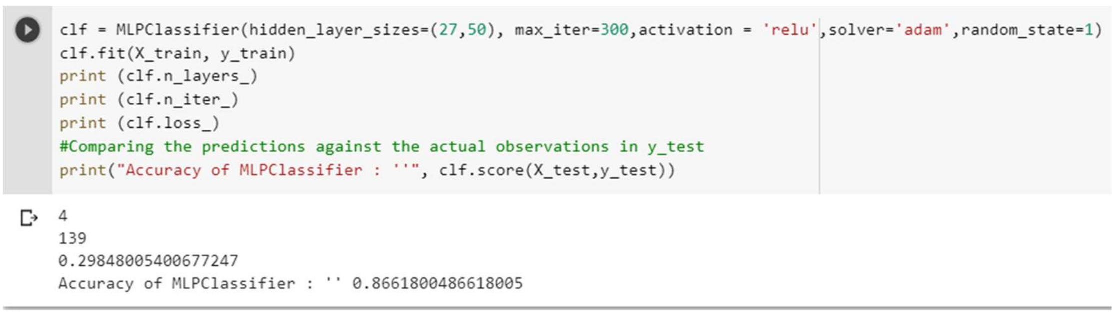
Subsequent work with multilayer perceptrons has shown that they are capable of approximating an XOR operator as well as many other non-linear functions. A multilayer perceptron is a class of feedforward artificial neural network. It consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function.

There are three common activation functions: sigmoids, hyperbolic tangent, and rectifier linear unit(ReLU). In recent developments of deep learning, ReLU is used more frequently as one of the possible ways to overcome the numerical problems related to the sigmoids. As a consequence, we use ReLU as the activation function in our MLP model.

Moreover, we are always looking to optimize model performance mostly by reducing the cost function associated with the model. There are several optimization algorithms that can help us improve model performance, including AdaGrad, RMSProp, and Adam, etc. In this project, we use Adam as the optimizer in the MLP model for some reasons. First, it's quite computationally efficient; second, it works well with large data sets and large parameters. What's more, it requires little memory space.

5.1.2 Parameter Setting

Under the condition of both reducing loss and increasing accuracy, we set different numbers of neurons and numbers of hidden layers with different feature selection methods, denormalized, and normalized dataset. For example, hideen_layer_sizes=(27,50) represents 27 neurons in first hidden layer and 50 neurons in second layer respectively. Under the circumstance, the loss is reduced to 0.298. Either adding or reducing a hidden layer will lead to greater loss. Fig. 6 gives the python code.



```
clf = MLPClassifier(hidden_layer_sizes=(27,50), max_iter=300,activation = 'relu',solver='adam',random_state=1)
clf.fit(X_train, y_train)
print (clf.n_layers_)
print (clf.n_iter_)
print (clf.loss_)
#Comparing the predictions against the actual observations in y_test
print("Accuracy of MLPClassifier : ''", clf.score(X_test,y_test))
```

```
4
139
0.29848005400677247
Accuracy of MLPClassifier : '' 0.8661800486618005
```

Figure 6. MLP model

5.1.3 Experiment Results

The accuracy of the MLP model using an accuracy score from sklearn with four feature selection methods is shown in Table 4.

Table 4. MLP experiment results

	All Features	Correlation	mRMR	RF
After normalized	0.862	0.866	0.860	0.867
Before normalized	0.875	0.702	0.873	0.875

We can find out that the accuracy obtained with mRMR and the normalized dataset is the worst (0.86). With the denormalized dataset, the accuracy obtained with correlation is only 0.702. Overall, RF has greater performance; especially with 10 features, the accuracy can be up to 0.87. As a consequence, we recommend use RF as feature selection methods with the MLP model.

5.2 Support Vector Machines (SVM)

5.2.1 Principle of SVM

SVM, an algorithm used for classification, is proposed by Vapnik and his colleagues according to statistical learning theory. SVM shows many advantages in solving the problems of the small sample and high-dimensional mode discrimination. SVM has been used in many practical problems like handwriting recognition, three-dimension target recognition, face recognition, and image classification.

In simple terms, the concept of SVM is to use the principle of the minimization of statistical risk to find a hyperplane to separate the two different sets and to maximize the margin between the two classes. For example, in two-dimensional space, the features are height and weight in Fig. 7. We want to classify all the data points to boys and girls. Since the data in this figure is not mixed together, they can be classified perfectly, and we called this situation hard-margin SVM. In the mathematical formula, boys ($y_i = 1$) should satisfy $w^T x + b \geq 1$ and girls ($y_i = -1$) should satisfy $w^T x + b \leq -1$ simultaneously. The goal of the SVM method is to find the hyperplane which can separate the two classes, and make the margins as bigger as possible.

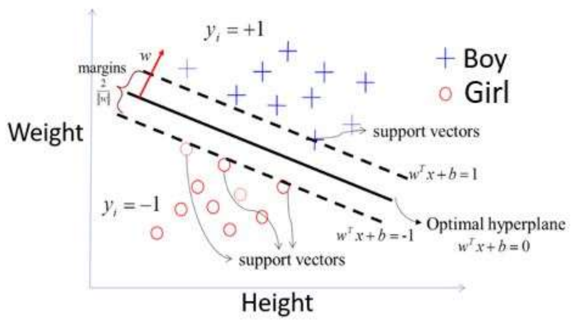


Figure 7. Hard-margin SVM

In reality, data is unlikely classified perfectly, so during the training process, some data can be tolerated to fall into the margins like the Fig. 8. shows, this situation we called soft-margin SVM.

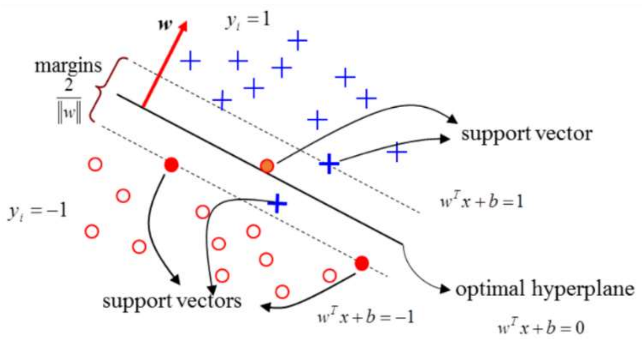


Figure 8. Soft-margin SVM

The method of tolerating the bias data is described below. For example, the lowest blue cross should fall within the range of $w^T x + b \geq 1$ theoretically, but due to the tolerance, the blue cross fall within the range of $w^T x + b \leq 1$. To match this problem, we only need to subtract a value from the original right formula, and this value called slack variable(ε), the parameter which tolerates data falling within margins. The bigger the value, the bigger the tolerant range.

If we want to obtain a perfect classified consequence, the choice of kernel function becomes very important. Kernel function can solve the problems when the data in different classes can’t be linearly separated in the original dimension by the method of non-linear projection to higher-dimensional space.

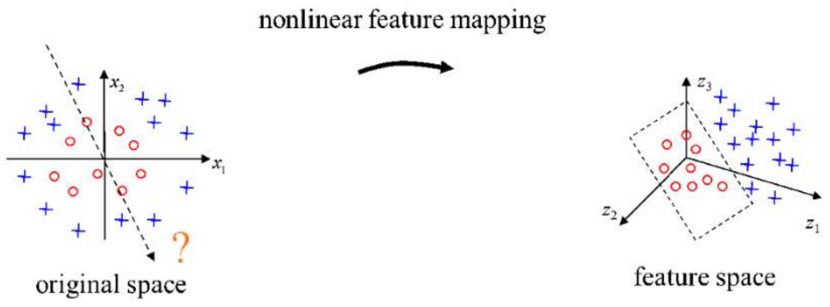
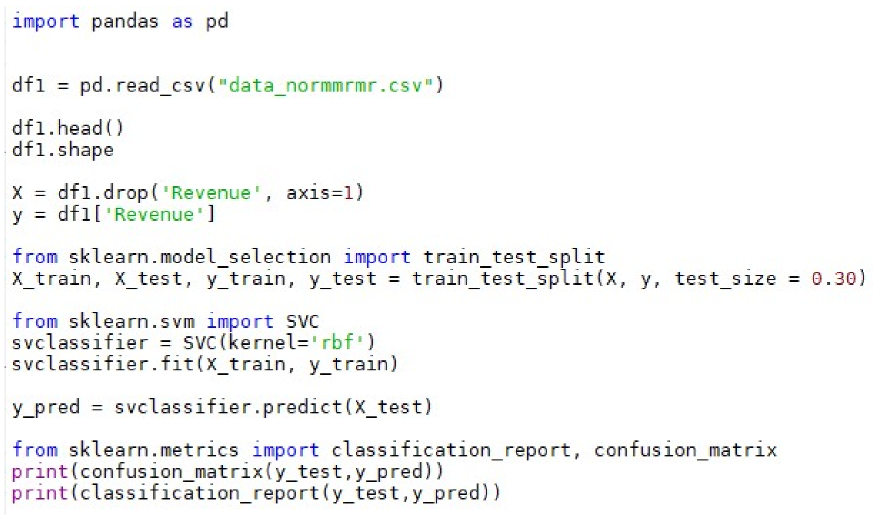


Figure 9. Map original space to higher dimension space

The original data can be separated into two classes by a hyperplane perfectly if it can be projected to a higher dimensional space called Hillbert space（H）in Fig. 9. The formula of projection is difficult designed, so we can use kernel function to help interpret it. The definition of kernel function is below, for all data, if a function can satisfy $\mathrm{k}(\mathrm{x}, \mathrm{y}) = < \varphi(\mathrm{x}), \varphi(\mathrm{y}) >$ k（x, y）is a kernel function, the content in the parenthesis is vector dot product, the model structure is showed in Fig. 10.



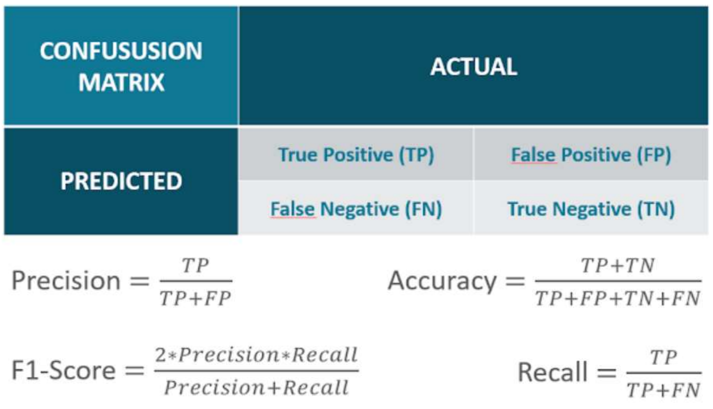
```
import pandas as pd


df1 = pd.read_csv("data_normmrmr.csv")

df1.head()
df1.shape

X = df1.drop('Revenue', axis=1)
y = df1['Revenue']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30)

from sklearn.svm import SVC
svclassifier = SVC(kernel='rbf')
svclassifier.fit(X_train, y_train)

y_pred = svclassifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

Figure 10. SVM model structure

5.2.2 SVM model construction

In the project, we first use a method called smote to obtain oversampling data to deal with the problem of the unbalanced dataset. Second, we use three methods contains random forest, correlation, and mRMR to select the important features for data processing. Then, we use SVM to predict whether the users on the Internet will buy the products or not. The consequence can be classified into two classes: buy it and do nothing.

Above is the python code we used. We use pandas to read the dataset and separate the column "Revenue" from other columns to be the predicted consequence. Then we use a python package called train test split whose main function is to divide the data into training and testing sets randomly. In this case, our training set is 70%, and the testing set is 30%. After this step, we use another package called SVC, and the kernel is RBF to train the model. Finally, we print the confusion matrix which contains the information about accuracy showed in Fig. 11.



CONFUSUSION MATRIX	ACTUAL	
PREDICTED	True Positive (TP)	False Positive (FP)
	False Negative (FN)	True Negative (TN)

$$\text{Precision} = \frac{TP}{TP+FP} \qquad \text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$$

$$\text{F1-Score} = \frac{2*Precision*Recall}{Precision+Recall} \qquad \text{Recall} = \frac{TP}{TP+FN}$$

Figure 11. Confusion matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. TP are cases in which we predicted yes, and they do. TN is which we predicted no, and they don't. FP is which we predicted yes, but they don't actually are. (Also known as a "Type I error"). FN is which we predicted no, but they actually do. (Also known as a "Type II error"). Precision is used to evaluate when we predict yes, how often it’s correct, and recall is when it’s actually yes, how often does the model predict yes. Accuracy is overall, how often the classifier is correct. F1-score is a weighted average of recall and precision.

5.2.3 Experiment analysis of SVM

We combine four feature selection results（all data, extra tree, random forest, correlation, and mRMR）with four data processing methods（do nothing, normalization plus oversampling, denormalization but oversampling, normalization but no oversampling）. The training and testing sets are divided by the proportion of 70%:30% and 80%:20%. From Table 5, we can find that the proportions of the two sets in this setting have similar results, so both proportions can be adopted. Between the data is normalized or not, the former one has better prediction accuracy in classification using the SVM method. However, the comparison of the original data and the oversampling data, we can find that using the original data to train the model can obtain better results. Table 5 gives the combination results.

Table 5. Experiment results of SVM

Data processing	Accuracy			
train70% test30%	All Features	Correlation	mRMR	RF
After normalized	0.85	0.83	0.84	0.84
After normalized	0.86	0.88	0.88	0.88
Before normalized	0.73	0.73	0.85	0.76
Before normalized	0.84	0.85	0.84	0.85
train80% test20%	All Features	Correlation	mRMR	RF
After normalized	0.86	0.84	0.85	0.85
After normalized	0.88	0.89	0.87	0.88
Before normalized	0.74	0.75	0.84	0.76
Before normalized	0.85	0.83	0.84	0.82
Data processing	F1-score			
train70% test30%	All Features	Correlation	mRMR	RF
After normalized	0.874	0.873	0.857	0.877

After normalized	0.929	0.932	0.932	0.935
Before normalized	0.807	0.819	0.805	0.860
Before normalized	0.920	0.914	0.917	0.910
train80% test20%	All Features	Correlation	mRMR	RF
After normalized	0.882	0.876	0.864	0.874
After normalized	0.936	0.930	0.933	0.936
Before normalized	0.809	0.830	0.811	0.861
Before normalized	0.918	0.912	0.917	0.921

5.3 RNN & CNN

In this section, we focus on recurrent neural networks (RNN) and convolutional neural networks (CNN) in further data processing, model construction, model performance improvement, training, and testing.

5.3.1 Data processing for RNN & CNN

In order to make the data suitable for using RNN and CNN models, this section deletes the feature "Browser" according to the feature selection results above to facilitate the conversion of the array data into grayscale images; then converts the oversampled data into $m \times n$ grayscale images. The result is shown in Fig. 12~Fig. 15. Finally, the picture data is normalized after divided by 255.

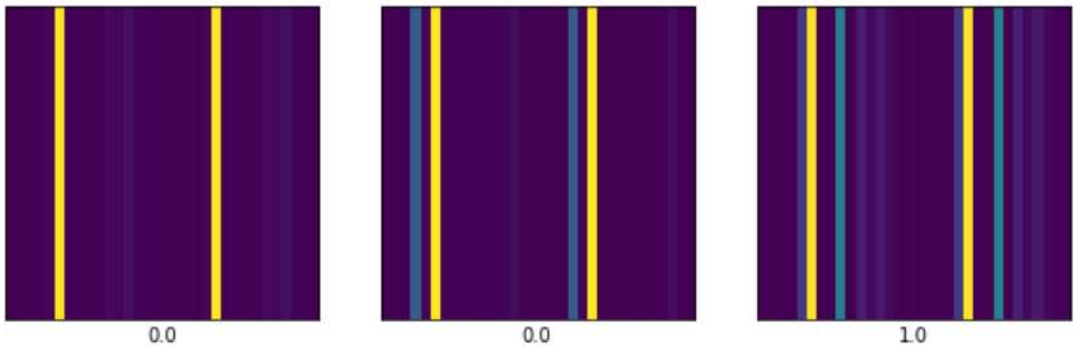


Figure 12. Grayscale image with 16 features

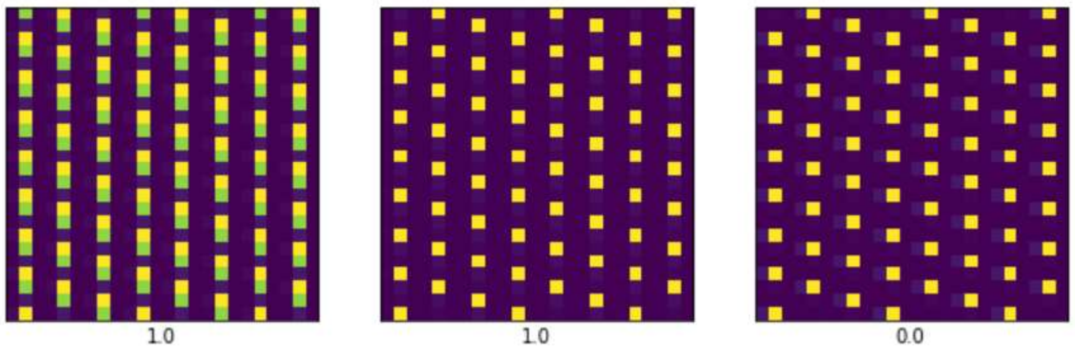


Figure 13. Grayscale image with 9 features selected by Pearson correlation

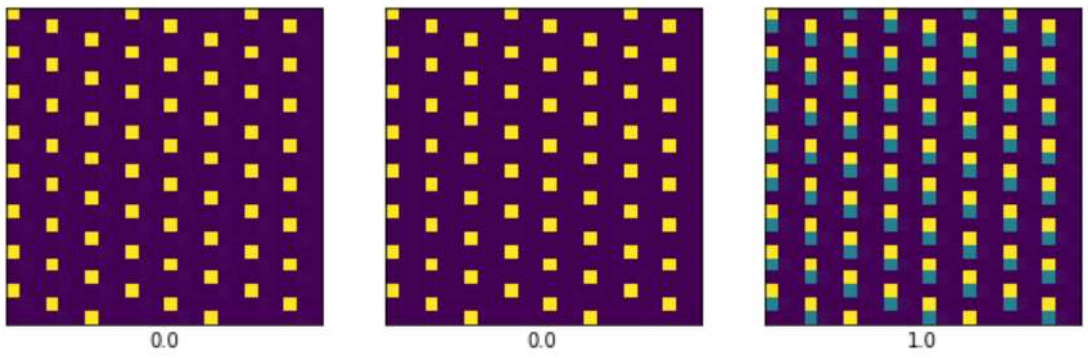


Figure 14. Grayscale image with 9 features by mRMR

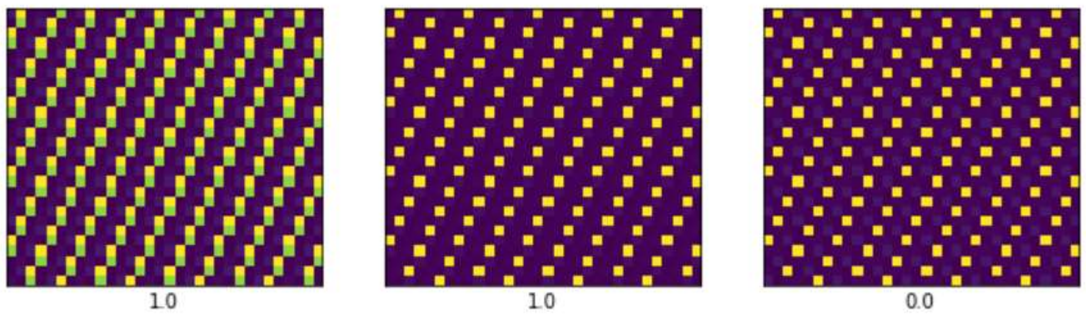


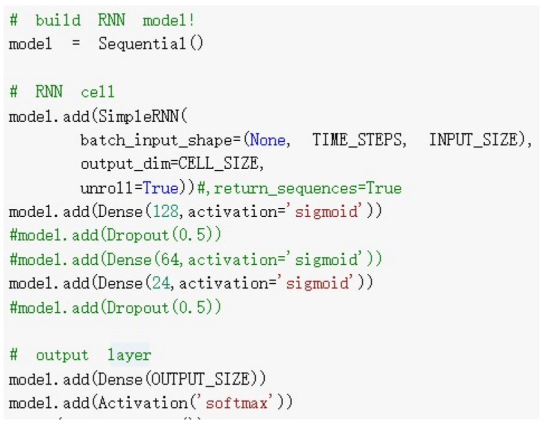
Figure 15. Grayscale image with 7 features selected by RF

5.3.2 Model architecture of RNN & CNN

The RNN and CNN model structures are shown in Fig. 16 and Fig. 17. Taking RNN as an example, the hyperparameters in the model are combined to obtain the optimal parameter setting level. The control variable method is used to adjust the parameters and perform experiments. We can see Table 6 for all the experimental results. As can be seen from the table, when the parameters of the RNN model are set to the parameter values of the bold part, the accuracy of the model prediction is the highest; the performance improvement method of the CNN model is the same, and will not be repeated here.

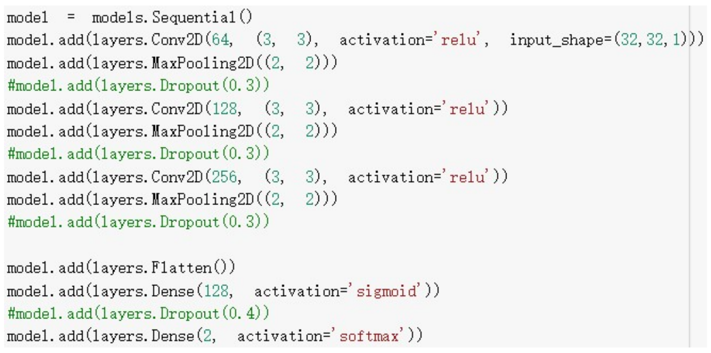
5.3.3 Training and testing for RNN &CNN

Fig. 18 and Fig. 19 give the program codes for RNN and CNN training. The number of training iterations is set to 20. The ratio of the training set to the test set of both models is 7: 3; RNN randomly extracts 20% of the data from the training set as the verification set, and CNN randomly extracts 12% of the training set as the verification set. The batch size of RNN training is set to 16 and that of CNN training is 32. The RNN monitors the value of "val_loss" during training and automatically reduces the learning rate to avoid falling into a bottleneck. The performance of RNN and CNN models with different features is shown in Table 7. There is a total of 8 experiments.



```
# build RNN model!
model = Sequential()

# RNN cell
model.add(SimpleRNN(
        batch_input_shape=(None, TIME_STEPS, INPUT_SIZE),
        output_dim=CELL_SIZE,
        unroll=True))#,return_sequences=True
model.add(Dense(128,activation='sigmoid'))
#model.add(Dropout(0.5))
#model.add(Dense(64,activation='sigmoid'))
model.add(Dense(24,activation='sigmoid'))
#model.add(Dropout(0.5))

# output layer
model.add(Dense(OUTPUT_SIZE))
model.add(Activation('softmax'))
```

Figure 16. RNN model structure

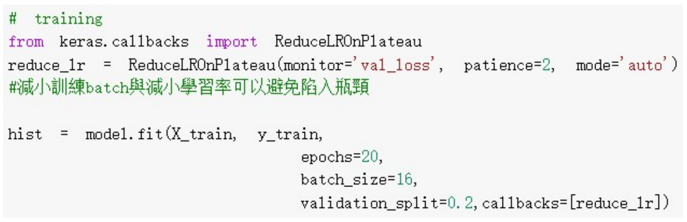


```
model = models.Sequential()
model.add(layers.Conv2D(64, (3, 3), activation='relu', input_shape=(32,32,1)))
model.add(layers.MaxPooling2D((2, 2)))
#model.add(layers.Dropout(0.3))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
#model.add(layers.Dropout(0.3))
model.add(layers.Conv2D(256, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
#model.add(layers.Dropout(0.3))

model.add(layers.Flatten())
model.add(layers.Dense(128, activation='sigmoid'))
#model.add(layers.Dropout(0.4))
model.add(layers.Dense(2, activation='softmax'))
```

Figure 17. CNN model structure

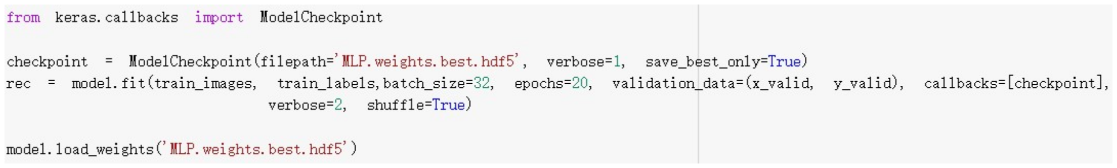
Table 6. RNN model parameter settings and experimental results

	Experiment No.	1	2	3	4	5	6	7	8
SimpleRNN	Layer Number	1	**1**	1	1	1	1	1	2
	Neuron Number	32	**32**	32	32	32	32	32	32/16
Other Algorithms	(Layers，Neuro Number)	0	**0**	0	0	0	0	LSTM(1, 16)	0
Dense	Layer Number	2	**2**	2	2	2	2	2	2
	Neuron Number	128/64	**128/64**	128/64	128/64	128/64	128/64	128/64	128/64
	Activation Function	relu	**sigmoid**	sigmoid	sigmoid	sigmoid	sigmoid	sigmoid	sigmoid
Output Layer	Activation Function	softmax	**softmax**	softmax	softmax	softmax	softmax	softmax	softmax
Learning Rate		0.001	**0.001**	0.001	0.001	0.001	0.001	0.001	0.001
Optimizer		Adam	**Adam**	SGD	RMSProp	AdaGrad	Adamax	Adam	Adam
Batch_size		16	**16**	16	16	16	16	16	16
Epoch		20	**20**	20	20	20	20	20	20
Accuracy	1th run	0.87051	**0.87456**	0.84455	0.87186	0.76075	0.86699	0.87105	0.87321
	2th run	0.86753	**0.87591**	0.78021	0.86834	0.75994	0.86807	0.87213	0.87024



```
#  training
from  keras.callbacks  import  ReduceLROnPlateau
reduce_lr  =  ReduceLROnPlateau(monitor='val_loss',  patience=2,  mode='auto')
#減小訓練batch與減小學習率可以避免陷入瓶頸

hist  =  model.fit(X_train,  y_train,
                             epochs=20,
                             batch_size=16,
                             validation_split=0.2,callbacks=[reduce_lr])
```

Figure 18. RNN training program code



```
from  keras.callbacks  import  ModelCheckpoint

checkpoint  =  ModelCheckpoint(filepath='MLP.weights.best.hdf5',  verbose=1,  save_best_only=True)
rec  =  model.fit(train_images,  train_labels,batch_size=32,  epochs=20,  validation_data=(x_valid,  y_valid),  callbacks=[checkpoint],
                             verbose=2,  shuffle=True)

model.load_weights('MLP.weights.best.hdf5')
```

Figure 19. CNN training program code

Table 7. The performance of RNN & CNN with different features

	All Features	Correlation	mRMR	RF
RNN	0.87537	0.87159	0.86780	**0.87780**
CNN	0.86834	0.86834	0.85104	0.87672

5.3.4 Generalizability of RNN & CNN model

According to overall experiment results (see code archive in the attachment), we can see that there is no overfitting in all trained models and the accuracy of the test set in RNN & CNN is higher than that of the training set and the validation set, reaching 87.54% and 86.83% respectively. In addition, the accuracy for testing becomes a little better after the feature selection (87.78% for RNN and 87.67% for CNN). Thus, we conclude that both the models of RNN and CNN have a good generalizability.

5.4 Comparison of experiment results

Table 8 shows the summary results of the above four model experiments. The results implicate that the SVM has the best performance after feature selection. Therefore, the SVM is the priority model to predict the on-line (real-time) shopping intention of customers.

Table 8. Testing accuracy of four models

	All Features	Correlation	mRMR	RF
MLP	0.862	0.866	0.860	0.867
SVM	0.860	**0.880**	**0.880**	**0.880**
RNN	**0.875**	0.872	0.868	0.878
CNN	0.868	0.868	0.851	0.877

6. Solutions for improving sale volumes

To achieve the sale goal (increase volumes by 20%), our website of the washing machine is designed to automatically offer content during a visit if the user is likely to abandon the site without shopping but predicted shopping intention is high. In other words, we apply our SVM model to predict purchasing intention after the website abandonment likelihood is more than a threshold and trigger the website to offer content for target customers, which imitate the behavior of a salesperson in the virtual shopping environment. In this way, we can timely take actions accordingly to improve the shopping cart abandonment and purchase conversion rates.

Besides, from the RF method, we can find the most important features including "PageValues", "ExitRates", "BounceRate", "Month", "ProductRelated", "ProductRelated_Duration", and "Administrative" in our dataset. Therefore, we try to propose some suggestions to improve the sales rate based on that. Both high "BounceRate" and "ExitRates" can represent an inappropriate design of the landing page and other pages of the website, so we need to redesign the content or the typesetting of the whole website to attract more customers. If the value of "ProductRelated" or "ProductRelated_Duration" is on the high side, it can represent that customers are interested in our products or they are hesitating. At this moment, we can encourage the customers to make the orders by sending additional advertisements or discounts automatically. Through the "Month" data, we can see that the sales rate is totally different from the slack season and peak season. So, our company can launch promotional activities to increase the sales rate in the slack season. All in all, we want to increase sales by about 20% to reach our target by the implementation of the above methods.

7. Conclusion and future work

In this project, we firstly construct a real-time purchasing intention prediction system for the virtual shopping environment, using aggregated pageview data kept track during the visit along with user information as input to machine learning algorithms. Besides, we apply oversampling and feature selection preprocessing techniques to improve the success rates and scalability of the algorithms and find out significant factors for purchasing intention. The experiments show that the SVM achieves the best results and the first 7 features selected by RF are important effect factors.

Based on the above work, we provide several solutions for improving the sale volumes of washing machines. First of all, the SVM model is applied with website abandonment likelihood prediction system simultaneously to trigger offering content during a visit, which imitates salesperson so that it enhances the purchase conversion rates and sale volumes. Sequentially, we give special suggestions according to 7 features selected by RF, respectively, such as redesigning the website, sending additional advertisements and so on.

However, machine learning models in this project are trained through data from the same source (only during one year). This means that the data used are consistent but influence the generalizability of models. Following that, we consider collecting more customer behavior data within several years since economic and technological environment changes brings noises to the data.

References

[1] Sakar, C. O., Polat, S. O., Katircioglu, M., & Kastro, Y. (2019). Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and LSTM recurrent neural networks. *Neural Computing and Applications*, 31(10), 6893-6908.

[2] Peng, H., Long, F., & Ding, C. (2005). Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (8), 1226-1238.