EE 559

MATHEMATICAL PATTERN RECOGNITION

PROJECT

DATA PRE-PROCESSING AND PATTERN CLASSIFICATION ON BANK MARKETING DATASET

Submitted by:
Nivedita Suresh
USC ID: 8602178289
Email: nsuresh@usc.edu
Date: 04-30-18

ABSTRACT

In this project we use Pattern Recognition Techniques to classify the Bank Marketing data set based on the features or attributes of the data. The data is processed to account for missing values and categorical data. Label Encoding and One Hot Encoding is used to process categorical data and the missing values are imputed by using the class mode method. The key features are selected from the normalized data using Principal Component Analysis or Fischer Discriminant Analysis and the data is sent for classification. Four types of classifiers are used to classify the training data and their performance is compared using performance metrics like classification accuracy, Area under ROC, F1 score and confusion matrix. The results are analyzed to determine which is the best parameter and classifier for classification of the data. The Naïve Bayes and SVM classifier gave the best results. Over-sampling the data to balance the class also helped improve the classification accuracy.

INTRODUCTION

The Bank Marketing dataset gives information of Bank clients, the Marketing strategies used and other attributes to finally predict if a client will agree to subscribe to a term deposit. The class label, thus, has only two outcomes; the client will either subscribe to a term deposit or the client will not subscribe to a term deposit. The goal is to predict the label or the outcome given a data with different client information and marketing strategies.

Pattern Classification involves the following steps:

- Measuring or acquiring the data
- Pre-processing the data
- Feature Extraction
- Classification
- Predicted class labels

For the pattern classification problem, we are already given the data; hence we can say that we are in the measurement space. Therefore we need to process the data, extract essential features from the data and classify the data. For classification, we train the data using three different classifiers; Support Vector Machine (SVM), Naïve Bayes Classifier and Mean squared error classifier.

In this project, I implement and discuss several Pre-processing and feature extraction steps that help improve the accuracy. The given dataset has a large number of samples that belong to one class and very small number of samples that belong to the other class. The project also discusses several ways that this problem can be overcome by using class weights or techniques like oversampling.

The classification performance is measured on the test set. For this, the Bank dataset is divided into two, in the beginning. The test set is kept aside. All pre-processing steps are applied to test step as well. The different performance measures that can be used to assess efficiency of the different classifiers are also studied in this project.

The Project is done on Python using Scikit-Learn toolbox.

PREPROCESSING

The first step of any classification problem is Pre-processing the given data. After importing the Bank marketing dataset, we perform a series of pre-processing routines that will give us a dataset where we have values in the pattern space.

Step 1: Identifying Unknowns
The bank dataset consists of several columns where many of the feature values are unknown. The unknown are identified and given the ASCII value '0', so that when we convert the feature into categorical values, they will be assigned an integer value 0. In this step we scan through the dataset to match if any feature has the string 'unknown' and replace this with '0'.

Step 2: Categorical Variables to Integers
The dataset consists of many features that are categorical strings. It is important to convert the categorical strings into integers. We do this using a Label Encoder function from the Scikit Learn tool box. The Label Encoder assigns integer values for all categorical strings based on their ASCII value. Any feature that contain unknowns or '0' will be assigned integer value 0. Other strings are assigned numbers in alphabetic order.
The final column is the class label; whether the client subscribes to the term of deposit. It takes string value 'yes' or 'no'. This will be converted into integer value 0 and 1; 0 for 'no' and 1 for 'yes'.

Step 3: Separating the Class label from the Data
The dataset contains all the data samples as well as the class labels. The last column of the Bank Marketing dataset is the class label. In this step, we remove the column containing the class label from the dataset and create two arrays that separately store the data set and the class label.

Step 4: Splitting Train and Test Data
Before we perform future pre-processing steps such as imputation of missing values and normalization, we need to separate the training and the test dataset.
The test data set is a novel dataset, never seen before by the classifier, which is used to measure the performance of classification. Therefore, we must make sure that the pre-processing operations do not use the test data or the test labels. Instead, all pre-processing steps are performed using training data, and the same transformation is extended to pre-process the test data. This ensures that no information about test data is known previously.

Step 4* (Optional): Oversampling

Oversampling is the process of generating data samples from classes that are under-represented. When number of data points belonging to one class is much larger compared to number of data points belonging to another class, we perform oversampling on the class with lesser number of elements so that the two classes are balanced. In this project, we use the Imbalance learn toolkit and the SMOTE function for oversampling. The Smote algorithm takes a data point belonging to the under-represented class and looks for its K-Nearest neighbours. It then creates new data point by interpolation. New samples are generated until the two classes are balanced. This is done only with the training dataset and helps to get better idea of the weights needed, to classify data points belonging to both classes accurately.

Step 5: Imputing Missing values in Training and Test Dataset

Imputing Training Data:

Missing values in training data are replaced by the class mode method. In this method we identify a missing data of a given feature and then look at the class label to which the data belongs. We scan through all training data of that particular feature with the same class label and assign the most frequently occurring feature value to the missing value. This is why the process is called Class Mode. The Imputer function in Scikit-learn replaces a missing feature value by the most frequently occurring feature value, irrespective of the class. Therefore, before we apply this method, we separate the training data into two arrays based on their class labels. Then we impute all missing features in each array, independently. After imputing missing values, we again combine the missing data to get the full training dataset.

Imputing missing values from the test dataset:

For the test dataset, we cannot use the above method. As explained earlier, we assume that the test data has not been seen before by the classifier and that the class labels are unknown to us. Therefore, we cannot use class labels to find the most frequent value. Instead we find the most frequent value in that feature column and use this to replace the missing value. This will be more incorrect than class mode method, but it will ensure that test data is not used for training.

Step 6: One Hot Encoding for Testing and Training data

The one hot encoding is the process of converting categorical integers into a one hot representation (Binary representation). For instance if we have a particular categorical feature for a dataset that takes 3 categorical value; 1,2,3 ; the one hot representation of these three categories is given by :

1 – [1 0 0]
2 – [0 1 0]
3 – [0 0 1]

It takes the total number of categorical strings for that feature (P) and creates P columns to represent that feature in a binary representation.

The Categorical representation is a very poor way of representing the features. Each category is represented by an integer. This causes categories represented by higher integers to be considered more important in some sense, as their magnitude is bigger. This is an incorrect way of representing features of equal importance or 'magnitude'.
One Hot Encoding is very useful in such cases where we do not want to give importance to any one categorical feature and we treat them all equally. In one hot representation above the distance between the 3 categorical integers 1, 2 and 3 are equal.
In some cases we do not use the one hot representation for categorical data. For instance if we have a categorical feature that denote different temperature 'cool', 'warm' and 'hot', and the categorical integer representation will be 0,1 and 2. We need not convert this into a one hot representation, because, 'hot' has a higher temperature magnitude than 'cool' or 'warm' and can be represented by a larger integer.
In our dataset, we use one hot representation for all the categorical data; Education, job, marital, default, housing, loan etc. The one hot representation transformation is computed for the training data and the same transformation is applied to test data so that the same categorical feature value in both data sets will be encoded in the same manner.

Step 7: Normalization

Sometimes datasets have some features that have very large magnitude while some features have very small values. It is very hard to compare the two features and understand their correlation when their magnitudes are so different. Normalizing features help in solving such problems. We calculate the mean and standard deviation of each feature in the training data and use this to normalize all the feature values in the training and test data.

$$\mu_f = \frac{1}{N}\sum_{i=1}^{N} x_{i,f}$$

$$\sigma_f = \frac{1}{N}\sum_{i=1}^{N}(x_{i,f} - \mu_f)^2$$

$$z_{i,f} = \frac{x_{i,f} - \mu_f}{\sigma_f}$$

Normalization is not performed on all features. The features represented by one hot encoding must not be normalized. In this project, normalization is performed on all numerical features.

The Pre-processing of the data is completed. The next step in Pattern classification is the feature extraction

FEATURE EXTRACTION

Feature extraction is the process of reducing the dimension of the feature space by selecting the best features, or the best linear combination of features that will improve classification. Again, the features to be extracted is computed only using the training data and the transform is applied to the test data. There are two ways in which we perform feature extraction in this problem; Feature selection and Feature reduction.

Feature Selection: Fischer Discriminant Analysis

In Fischer Linear discriminant Analysis, we select those features that result in high inter-class variability and low intra-class variability. This means that such feature will give low variances for all elements belonging to same class and high difference between elements belonging to other classes. The cost function is defined as:

$$J(w) = \frac{|(\widetilde{m_1} - \widetilde{m_2})|^2}{\widetilde{s_1}^2 + \widetilde{s_2}^2}$$

The D' features are selected such that they maximize the cost function.

Feature Reduction: Principal component Analysis

Feature reduction is the process of projecting the data from the existing high-dimensional feature space into a reduced space known as the principle component axis. Principal Component Analysis transform is computed by computing the Eigen vectors of the covariance matrix and selecting those corresponding to the dominant Eigen values.

$$\sum = \frac{1}{N}\sum_{j=1}^{N}(\underline{x_i} - \underline{m})(\underline{x_j} - \underline{m})^T$$

In this Project, we compare the results obtained using Principle Component Analysis with that of Fischer Discriminant Analysis to see which feature extraction method gives better classification results for Bank Marketing dataset.

CLASSIFICATION

The next step is the classification. Classification is performed on four different classifiers;

- **Perceptron Classifier** - Sklearn SGD Classifier
- **Support Vector Machine** - Sklearn SVM –SVC
- **Naïve Bayes Classifier** - Sklearn Gaussian NB
- **Random Forest Classifier** - Sklearn Random Forest Classifier

Here Naïve Bayes classification is a statistical classifier, while Perceptron classifier, Support Vector Machine and Random forest classifier are all distribution free.

CROSS-VALIDATION

For the classifier, there are several parameters that we can control. For instance, for the support vector machine we can control C and gamma. Some choices of parameters leads to overfitting of training data and some choice of parameter results in very poor classification of the training data. In both cases this will affect the test data set accuracy. Therefore, it is important that we select the right parameters that give good classification accuracy while not overfitting the training data. This is why we need a validation set. The validation set is a dataset taken from the training dataset and kept aside for computing the best parameters to use for the classifier. In this project we perform cross-validation. In cross validation, we take an array with different choices of parameters. We select one parameter from this array and train the data. The training set is divided into k folds. One of the k blocks is kept aside as the validation set and the other k-1 blocks are used for training using the parameter we selected. We get the validation accuracy. Now we randomly use another block out of the k blocks as the validation block and repeat the procedure. This is known as the round robin cross validation. The mean of all the accuracies are taken as the average accuracy for that parameter. This is repeated for all parameters in the array and the parameter that gives the highest validation accuracy is used as the optimal one. This gives much better results than using the default parameter or randomly initializing a parameter.

In this project, cross validation was performed for SVM classifier to select optimal C and gamma and Random forest classifier to select optimal depth of decision tree.

PERFOMANCE PARAMETERS

The Performance is measured using the following performance measure:

- **Classification Accuracy:**
  Classification Accuracy is calculated as the ratio of the total number of samples classifier correctly to the total number of samples.

  $$\text{Classification Accuracy} = \frac{Number\ of\ samples\ classified\ correctly}{Total\ number\ of\ samples}$$

- **Area Under Curve for ROC curve:**
  Receiver operating characteristic curve, for a binary or two-class classifier, is the graph between the true-positive rate and the false-positive rate for different threshold used for prediction. The true positive rate (TPR) is the ratio of the number of true positive to the total number of actual positives. False positive rate (FPR), similarly, is the ratio of the number of false positive to the total number actual negatives.
  The curve is obtained for different values of TPR and FPR by varying the threshold. Ideally the TPR must be greater than FPR. This means that the classifier will give a higher probability for a randomly selected positive data point than a randomly

selected negative data point. When TPR =FPR, the probability scores are the same and it is like a random guess.
The AUROC is the area under the ROC. It must be greater than 0.5 for TPR>FPR.

- **F1 Score:**
  The F-score is obtained by calculating the harmonic mean of the precision and recall. It must be very high for good classification.

$$F\text{-}score = \frac{2*Precision*Recall}{Precision+Recall}$$

- **Confusion Matrix:**
  It is a table that gives the number of true positive, true Negative, false positive and false negative.

RESULTS

Parameter Comparison

Parameter	SVM	Perceptron	Naïve Bayes	Random Forest
No oversampling No FDA No PCA	Test Acc: 0.888 F1 score: 0.0 AUC: 0.498	Test Acc:0.8582 F1 score: 0.24048 AUC: 0.5877	Test Acc: 0.8009 F1 score: 0.27 AUC: 0.597	Test Acc:0.88349 F1 score: 0.0 AUC: 0.4959
No Oversampling No FDA PCA	Test Acc: 0.8907 F1 score: 0.0 AUC: 0.5	Test Acc: 0.4815 F1 score: 0.18 AUC: 0.499	Test Acc: 0.7888 F1 score: 0.287 AUC: 0.6134	Test Acc: 0.882 F1 score: 0.0 AUC: 0.495
No Oversampling FDA PCA	Test Acc: 0.6145 F1 score: 0.21386 AUC: 0.55464	Test Acc: 0.73009 F1 score: 0.13 AUC: 0.49150	Test Acc: 0.3213 F1 score: 0.1747 AUC: 0.46756	Test Acc: 0.8563 F1 score: 0.051 AUC: 0.4964
Oversampling No FDA No PCA	Test Acc: 0.8631 F1 score: 0.2459 AUC: 0.57396	Test Acc: 0.7864 F1 score: 0.3820 AUC: 0.7054	Test Acc: 0.895145 F1 score: 0.25 AUC: 0.5725	Test Acc: 0.8669 F1 score: 0.17964 AUC: 0.54510
Oversampling No FDA PCA	Test Acc: 0.8699 F1 score: 0.20238 AUC: 0.5545	Test Acc: 0.82524 F1 score: 0.2857 AUC: 0.60313	Test Acc: 0.8359 F1 score: 0.2683 AUC: 0.5897	Test Acc: 0.8359 F1 score: 0.242 AUC: 0.5742
Oversampling FCA PCA	Test Acc: 0.86602 F1 score:0.28125 AUC: 0.59111	Test Acc: 0.80097 F1 score: 0.373 AUC: 0.68649	Test Acc: 0.86796 F1 score:0.3645 AUC: 0.6387	Test Acc: 0.851 F1 score: 0.295 AUC: 0.60233

Best Result on Classifier:

Support Vector Machine:

SVM CLASSIFIER

```
Training Accuracy using SVM Classifier:  0.8925845147219194
Training AUC using SVM Classifier:  0.8925845147219194
Training F1 SCORE using SVM Classifier:  0.8833629366489045

             precision    recall  f1-score   support
        0.0       0.84      0.97      0.90      2751
        1.0       0.97      0.81      0.88      2751
avg / total       0.90      0.89      0.89      5502

CONFUSION MATRIX:
[[2673   78]
 [ 513 2238]]

Testing Accuracy using SVM CLassifier:  0.8699029126213592
Testing AUC using SVM Classifier:  0.554501500661063
Testing F1 SCORE using SVM Classifier:  0.2023809523809524

             precision    recall  f1-score   support
        0.0       0.90      0.96      0.93       917
        1.0       0.31      0.15      0.20       113
avg / total       0.84      0.87      0.85      1030

CONFUSION MATRIX:
[[879  38]
 [ 96  17]]
```

Perceptron Classifier:

```
PERCEPTRON CLASSIFIER

Training Accuracy using PERCEPTRON Classifier:  0.7064703744093057
Training AUC using PERCEPTRON Classifier:  0.7064703744093057
Training F1 SCORE using PERCEPTRON Classifier:  0.6879227053140097

             precision    recall  f1-score   support
        0.0       0.68      0.77      0.72      2751
        1.0       0.73      0.65      0.69      2751
avg / total       0.71      0.71      0.71      5502

CONFUSION MATRIX:
[[2107  644]
 [ 971 1780]]

Testing Accuracy using PERCEPTRON CLassifier:  0.7864077669902912
Testing AUC using PERCEPTRON Classifier:  0.7054651084239681
Testing F1 SCORE using PERCEPTRON Classifier:  0.38202247191011235
             precision    recall  f1-score   support
        0.0       0.94      0.81      0.87       917
        1.0       0.28      0.60      0.38       113
avg / total       0.87      0.79      0.82      1030

CONFUSION MATRIX:
[[742 175]
 [ 45  68]]
```

Naïve Bayes:

```
NAIVE BAYES CLASSIFIER

Training Accuracy using NAIVE BAYES Classifier:  0.717739003998546
Training AUC using NAIVE BAYES Classifier:  0.717739003998546
Training F1 SCORE using NAIVE BAYES Classifier:  0.6484038940457323

             precision    recall  f1-score   support
        0.0       0.66      0.91      0.76      2751
        1.0       0.86      0.52      0.65      2751
avg / total       0.76      0.72      0.71      5502

CONFUSION MATRIX:
[[2517  234]
 [1319 1432]]

Testing Accuracy using NAIVE BAYES CLassifier:  0.8679611650485437
Testing AUC using NAIVE BAYES Classifier:  0.638760482913695
Testing F1 SCORE using NAIVE BAYES Classifier:  0.36448598130841126

             precision    recall  f1-score   support
        0.0       0.92      0.93      0.93       917
        1.0       0.39      0.35      0.36       113
avg / total       0.86      0.87      0.86      1030

CONFUSION MATRIX:
[[855  62]
 [ 74  39]]
```

Random Forest Classifier:

```
RANDOM FOREST CLASSIFIER

Training Accuracy using Random Forest Classifier:  0.9881861141403127
Training AUC using Random Forest Classifier:  0.9881861141403127
Training F1 SCORE using Random Forest Classifier:  0.9880930573365085

             precision    recall  f1-score   support
        0.0       0.98      1.00      0.99      2751
        1.0       1.00      0.98      0.99      2751
avg / total       0.99      0.99      0.99      5502

CONFUSION MATRIX:
[[2740   11]
 [  54 2697]]

Testing Accuracy using Random Forest Classifier:  0.8514563106796117
Testing AUC using Random Forest Classifier:  0.6023344688817903
Testing F1 SCORE using Random Forest Classifier:  0.29493087557603687
             precision    recall  f1-score   support
        0.0       0.91      0.92      0.92       917
        1.0       0.31      0.28      0.29       113
avg / total       0.85      0.85      0.85      1030

CONFUSION MATRIX:
[[845  72]
 [ 81  32]]
```

DISCUSSION

The best accuracies for all the classifiers were compared. We find that oversampling gives the best results. Oversampling adds data points to the under-sampled class. This helps in actually understanding how different features are correlated for the under-represented class label. It also resolves the issue of class imbalance.
We also see that using Principal component analysis is better than using Fischer Discriminant Analysis for this problem. This is due to the fact that there are many categorical data, which results in all those features being equidistant from each other. This makes it difficult to understand which choice of features increases the distance between the two class means.

When oversampling was not used, we observe that the classifiers tend to classify all the data points of the majority class accurately, while getting very high inaccuracies for the under-represented class. This is due to the data imbalance. This is why we use the F1 score and AUC performance measure to ensure this does not happen. This problem can be fixed to some extent by using class weights while fitting the classifier.

The SVM and Naïve Bayes classifier gives the best accuracies, while the Perceptron model gives the worst accuracy. The Random Forest classifier is found to give very high accuracy but result in low F1-score as it tends to favour the oversampled class. The Naïve Bayes classifier gives good results without oversampling as the priors take care of the class weights. This improves the AUC and the F1 score.

The F1-scores are found to be between 0.2 and 0.4. We can get a measure of the weighted F1-score to determine whether the classification is good or not. This will give a higher value than the F1-score, as it takes into consideration the difference in the number of class objects.

In conclusion, we can say that Naïve Bayes and SVM classifier both work well in classifying this dataset.

SOFTWARE AND TOOLS USED FOR THE PROJECT

- Python 3.5
- Scikit-Learn
  - sklearn.preprocessing
  - sklearn.feature_selection
  - sklearn.decomposition
  - sklearn.model_selection
  - sklearn.svm
  - sklearn.naive_bayes
  - sklearn.ensemble
  - sklearn.linear_model
  - sklearn.metrics
  - feature_extraction
- NumPy
- Pandas
- Imbalance-Learn
  - imblearn.over_sampling