Implementation and Analysis of An Efficient CRM-Data Mining Framework

Chao Pi, Xunan Dai, Ryan Haoyuan Wang

**Abstract**—Customer relationship management (CRM) has a simple goal that is to boost business by improving customer relationships with the firm. A CRM system can enhance a company’s relationships and interactions with customers if utilized positively. The research paper this report is referring to proposed a framework that can process and predict customer’s decision on a business campaign. Their approach followed a basic data mining procedure with data preprocessing, model building, and model evaluation. This report is an extension to their work to perfecting their framework by exploring more possible methods in the data mining process. By applying several new changes in the process, hidden knowledge and information can be extracted for companies to select the most loyal customer with best-fit products and services.

**Index Terms**—CRM, Data Mining, Machine Learning, Classification, Naïve Bayes, MLPNN, SVM, K-NN, Decision Tree

—————————— ◆ ——————————

# 1 Introduction

THIS report is an extension of an implementation of a data mining framework from the significant paper *An Efficient CRM-Data Mining Framework for the Prediction of Customer Behavior* which proposed a framework that can predict customers’ decision on campaigns from financial institutions by building models on the given dataset. Two classification models: Multilayer Perceptron Neural Networks and Naïve Bayes were used for comparison in the study, and the result was concluded by the performance and accuracy that MLPNN is better for the framework because it has a better accuracy. Since the conclusion from the study only considered accuracy for models’ performance, it cannot reveal the hidden knowledge of the dataset to build the best performing framework for CRM systems. This report explores more approaches to enhance this proposed framework to an extent. The dataset analysis will start by investigating the distribution of different attributes like age, occupation and bank savings which leads to a more sophisticated method on dataset preprocessing. Prediction of missing values and value imputation are performed to improve the framework’s design. Several more classification models are considered to be applied to compare the result with. The models include Support Vector Machine, Decision Tree and K-Nearest Neighbor. The performance of these classification models is investigated in this report, and the hidden knowledge can be drawn from the models and their analysis.

# 2 Dataset Analysis

## 2.1 Dataset Introduction

The dataset is acquired from UCI machine learning repository. It contains data related to a Portugueses bank’s marketing campaigns information related to the customers and their decision on the campaign (yes or no) from 2008 to 2010. The campaign is a long-term deposit application that the bank offers to customers as a product. The campaign is conducted by phone calls; therefore, the dataset contains information regarding the phone call’s date, duration, and number of previous communication contacts with the customer. The intention for this report is to make accurate classifications on the customers’ final decisions based on their information acquired from the dataset.

## 2.2 Feature Introduction

The dataset consists of mainly customers’ personal and financial information as the attributes of this Portuguese bank. There are previously 58 columns, and with sensitive private information removal, a total of 20 attributes and 1 target output are left for investigation. There are 7 attributes that are related to customer’s personal and financial information including age, job, marital status, education level, if has credit in default and if has loan on housing. There are 4 other attributes related to the customer’s relationship with the bank constituted by the information of last contact of current campaign including contact communication type, last contact month of year, last contact day of week and last contact duration in seconds. There are another 9 attributes related to the customer such as number of contacts, the previous outcomes, employment variation rate and consumer confidence index etc., The final target result that the customer responds to the campaign has 1 variable indicating if the customer agreed or declined the campaign.

## 2.3 Numerical Attributes

For each customer’s personal information, only their age is represented by numeric values. The other important numerica features includes last contact duration in seconds, number of contacts for current campaign, days passed since last communication and different rate represents the banks’ relation with the customer.

## 2.4 Categorical Attributes

Besides the above stated attributes, the rest all are categorical features that essentially represent the customer’s personal, financial and educational status. Each categorical features label indicates the customer’s potential to finally agree on the campaign.

## 2.5 Dataset Visualization

Before proceeding to dataset preprocessing, the dataset should be plotted to provide a general view of the information from this dataset.

First, a pie chart (Fig.1) is plotted to illustrate the percentage of customers that agreed to take the deposit application. It can be seen from the plot below that only 11% of the customers said yes to this campaign; wheres, 89% of customers refused this certain campaign.

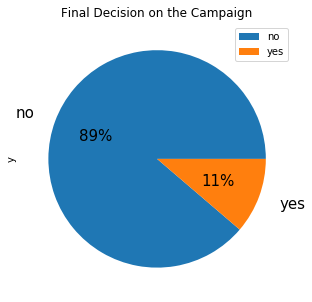
****

Fig. 1. Percentage of the customer agreed and declined on the campaign.

Therefore, a closer look to which part of the population in the contacted customers agreed to this campaign can gain important information for this bank's future business regarding customer relationship. Below is a stacked bar chart combined with line plot (Fig.2) shows the percentage of each age that the distribution of final decision customers made. It can be observed that the apparent distribution of age below 20 and age above 60 has the highest percentage rate accepting the campaign with taking the bank’s deposit application. However, most age is located in range of 20 to 60, which causes the lower acceptance rate for this campaign.

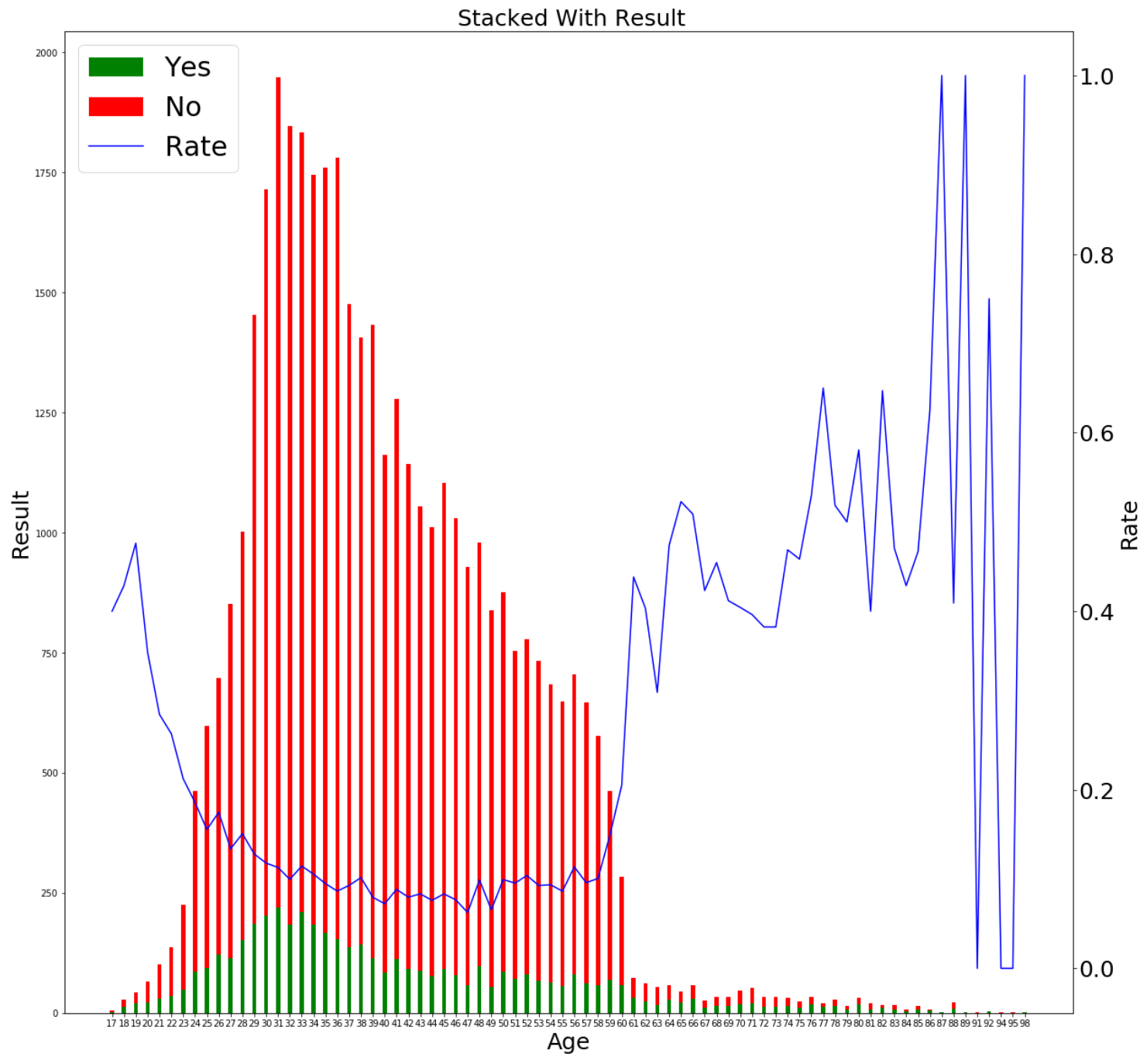


Fig. 2. Age distribution of customers showing their final decisions on the campaign.

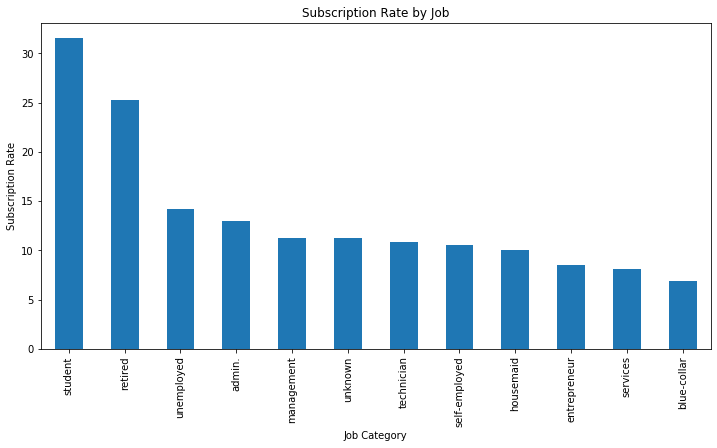


Fig. 3. The bar chart of campaign acceptance rate per job category.

Another view to visualize this customer dataset is to sort their campaign acceptance rate by their job categories. Interestingly, the plot (Fig.3) shows that the group of students has the largest percentage accepting the bank’s deposit campaign. This just correlated to previous findings of the age distribution that age below 20 has a higher acceptance rate. Following the students are the group of retired whom has the second highest acceptance rate that corresponds to the previous finding.

From the dataset description, it states the duration of the last contact can have a big effect on the output result, so it should be handled with care for predictive models. The following plot (Fig.4) shows the relationship between the duration of calls in current campaign and the number of past contacts with generated campaign results. It can be seen that the bigger the number of contacts the bank have made, the bluer the plot has which means the customer gets annoyed by the marketing phone calls that leads to campaign failure. An average of 5 time contacts can generate a good campaign result on a customer.

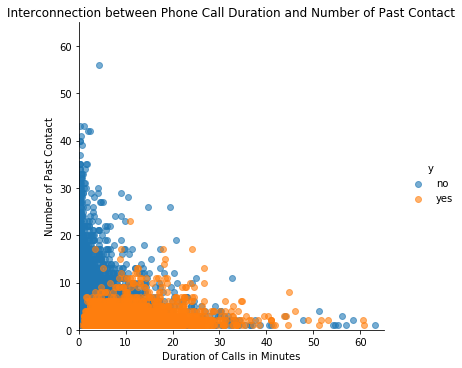


Fig. 4. The interconnection between phone call duration and number of past contact. The final decision is plotted with different color.

Attributes correlation map (Fig.5) is another perspective that can illustrate the relationship among each pair of attributes. For most features, there are no obvious correlations. However, the lower right corner is a special case, which is analysed in data reduction section. There is one attribute that needs to pay special attention to in the correlation map is the duration of the phone call that positively correlated to the final result. Per dataset description, duration attribute needs to be dropped if the model is built for realistic predictions.

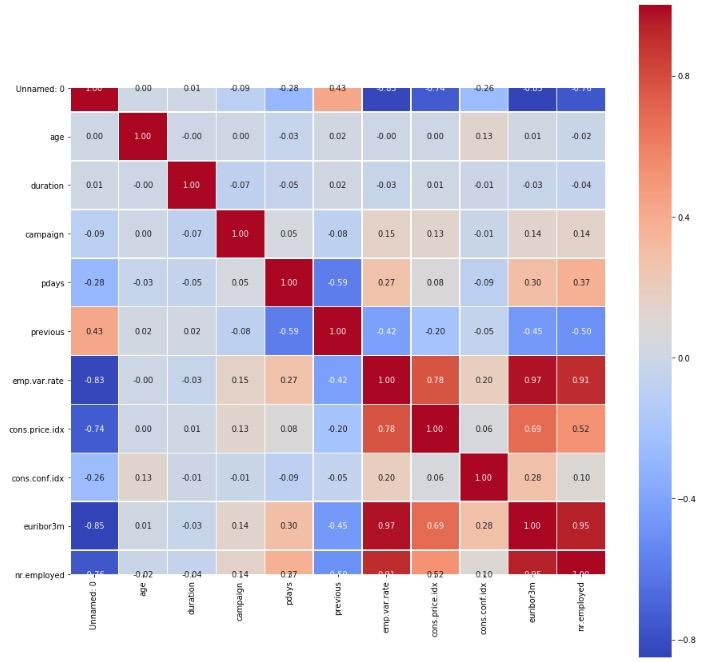


Fig. 5. The correlation map shows the correlation coefficient of each pair of numerical attributes.

# 3 Dataset Preprocessing

Dataset preprocessing is an essential step in data mining framework. It has an important role to process the data for obtaining a better result from model building. In CRM framework, figuring out which attributes that has the most influence on the customer’s final decision is the key to identify target points for future campaigns.

There are several steps involved in data preprocessing, and this report focuses on dealing with missing values by imputation, removing outliers and correlation analysis for feature reduction.

## 3.1 Missing Values

According to the dataset description, the dataset acquired from Portugese bank has several attributes contain labels marked as “unknown” that can be processed by recognizing as labeled class, value imputation or simply deletion. By merely glimpse of the dataset, there are numerous “unknown” sparsely spread in different attributes. An approach is needed to properly deal with these data blocks.

The counted number of “unknown” values involves attribute as follows: 'job': 330, 'marital': 80, 'education': 1731, 'default': 8597, 'housing': 990, 'loan': 990. All of the attributes with missing values are categorical features means that methods like mean-value as well as the mode-value imputation for the “unknown” values cannot be used. Therefore, this report uses the method of predicting and imputing the missing values with various algorithms.

The simplest approach to deal with the missing values is to treat “unknown” blocks as a labeled class, and hence make predictions from the models generated by them. This approach is tested in the following section and the result is shown for evaluation.

The approach this report used is to treat these “unknown” blocks as missing values, and hence predicting these missing values using Decision Tree or Naïve Bayes. The attributes that carry missing values are treated as target field. The tuples without missing values are used as training set, ann tuples with missing values are applied with trained models for prediction.

The last approach is to drop the tuples contain these “unknown” values. This approach is used in the significant paper and their result is used as base-line implementation comparison. Considering that dropping these tuples may cause information lost since the size of the dataset is not big, predict imputation methods are implemented in order to compare each model’s with different data preprocessing approaches.

## 3.2 Outliers Analysis

Outliers are data points in a attributes that has abnormal values by comparing with the average in this attribute. Outliers have great effects on the final decision made by prediction models since their abnormal values; therefore, affecting the generated model’s performance. To find out the presence of outliers in the dataset, the boxplots of each numeric attribute are plotted below (Fig.6, 7).

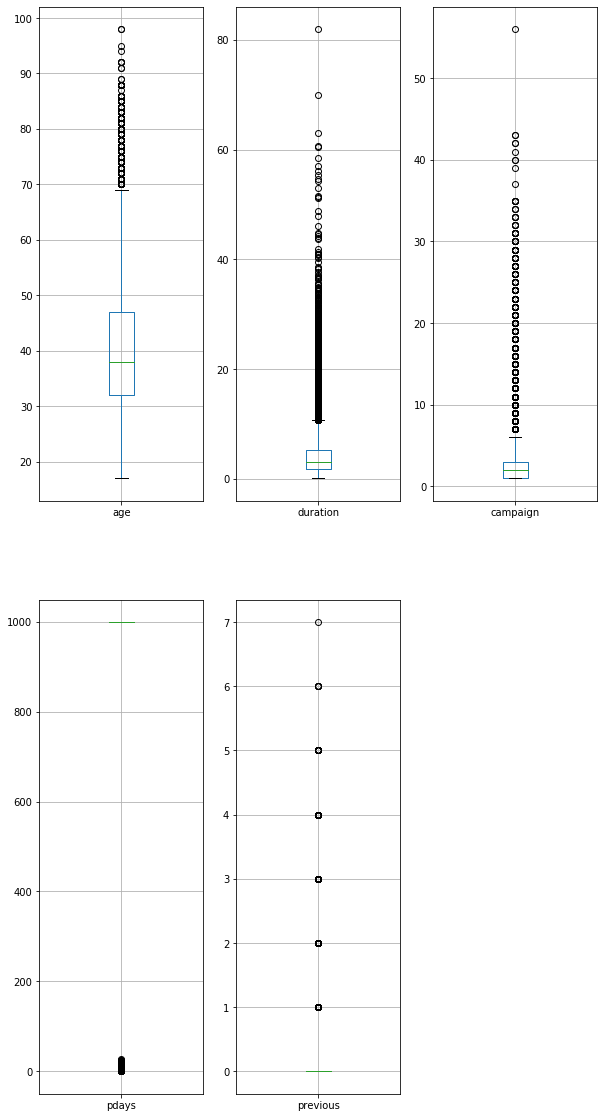
****

Fig. 6. Box plot of numeric attributes: age, duration and campaign for outlier analysis.

The first subgraph plotted in fig.6 is the age distribution among all the customers. The average age in this dataset is around 32 to 47. Customers with age above 70 are marked as outliers. However, these customers’ age cannot be treated as outliers since their age is not affecting their final decisions on the campaign, and it is common sense that human beings can have an age above 70.

The second subgraph plotted in fig.6 is a box plot of phone call duration of current campaign that bank had with the customers. Outlier handling is urgent in this attribute since there are many outliers that present in the dataset, and the fact is that this attribute contributes a great amount to the customers’ final decision. This attribute should be handled with care during the data preprocessing step. If the goal is to build model for realistic prediction, this attribute should be dropped since the duration is not known before the final decision is made by the customer. Otherwise, the outlier should be analyzed case by case since they represent some special cases that customers may need more persuasion or they have special needs to fulfill in order to accept the campaign. In this report, duration feature is dropped in order to make the system more suitable for realistic environment.

The third subgraph in fig.6 shows the number of communication the bank had for this current campaign. Most customers was contacted by the bank below 4 times, and the average is at about 2 times. Whereas, there are outliers above 6 times, and at one extreme one customer has being contacted by the bank with 58 times. These clients may need more investigation, and they may present a great value to the final prediction. Comparison will be made with removing these outliers and keeping these outliers.

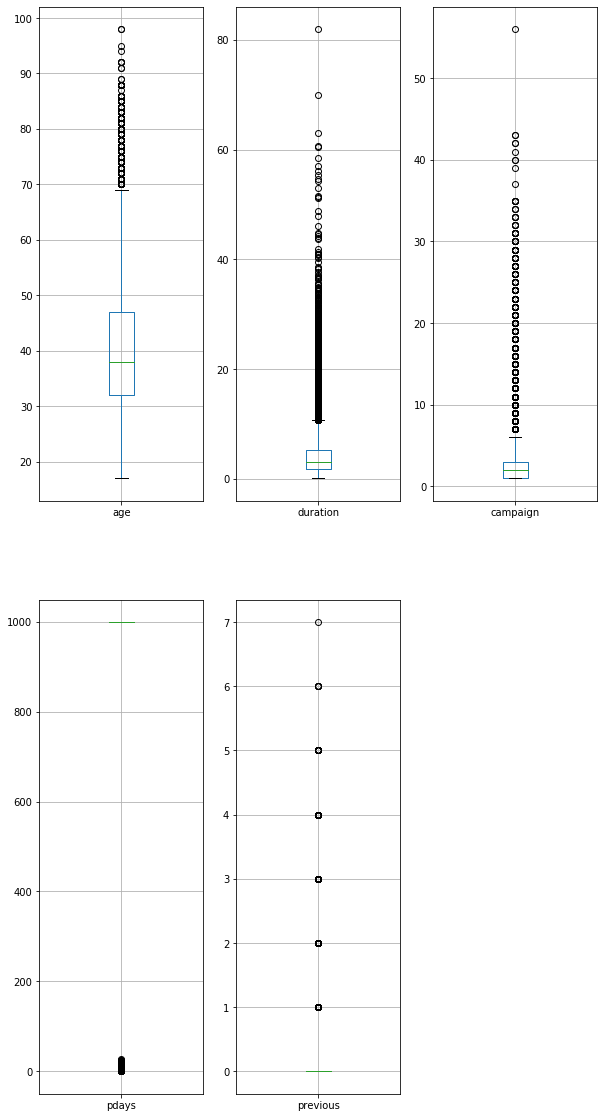
****

Fig. 7. Box plot of numeric attributes: pdays and previous for outlier analysis.

The two box plots in fig.7 represent the days past since last communication with the customer and the number of communication for previous campaign. Their outliers do not have direct relation to the final decision of the customer, and they do not alter the algorithm processing since they represent real conditions, so they are kept for later model implementation.

## 3.3 Correlation Analysis

From the correlation map, the lower right corner is shown (Fig.8) as follow that several features are highly correlated. These features are: 1) emp.var.rate and cons.price.idx that have correlation coefficient at 0.78; 2) emp.var.rate and euribor3m that have correlation coefficient at 0.97; 3) emp.var.rate and nr.employed that have correlation coefficient at 0.91; 4) euribor3m and nr.employed that have correlation coefficient at 0.95.

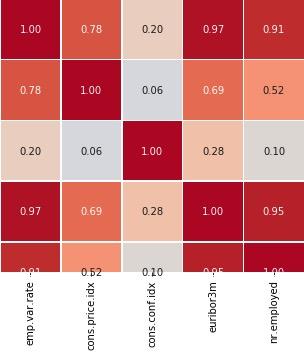


Fig. 8. The lower right corner highly correlation features.

## 3.4 Feature Reduction

To reduce the repeat computation during the model building phase, features with repeated information have to be removed from the dataset. Feature selection, feature importance analysis and principal component analysis are applied in this report to realize feature reduction.

First, since the highly correlated features are not helpful for the model building and for later customer behavior’s classification, two attributes are selected representing the five features: emp.var.rate and cons.conf.idx. Three attributes: cons.price.idx, euribor3m and nr.employed are dropped to reduce the feature complexity.

Next, by applying Scikit-learn package’s extra-tree classifier, the top-10 important features are selected and plotted (Fig. 9) based on the feature importance in order to help decide the number of components of PCA.

Principal component analysis is an approach by applying orthogonal transformation to represent maximum information of a dataset and reduce the dimensionality of the dataset in the meantime. Therefore, PCA is implemented and in order to cover as much information as possible, the final PCA uses 5 components to represent the dataset.

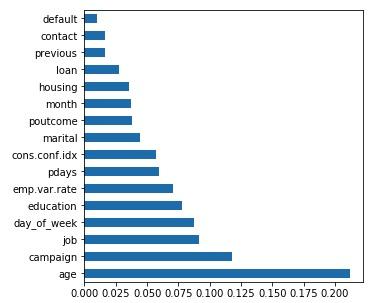


Fig. 9. Top-10 important features for PCA analysis.

# 4 Model Implementation

In order to have a generalized result, several different classification models are picked and implemented in this report. They are compared for their performance based on the metric of accuracy. The selected classification models includes Multilayer Perceptron Neural Network, Naïve Bayes, Support Vector Machine, K-Nearest Neighbor and Decision Tree. These models are all suitable for classification prediction, and their pros and cons are analyzed in the following sections.

## 4.1 Neural Network

Neural network is considered as one of the most powerful classification algorithms among machine learning models, which can get good results in high dimensional data. It has been widely used in industry and has many existing APIs. In this report, MLPClassifier is applied from Scikit-learn’s neural network package. Though the neural network can always generate good results from the training data, the training complexity is usually higher than other models. This model may take longer time and more computation resources to train. The numerous and complex parameters of neural network are also hard to tuning. In this report, the MLPNN classifier is constructed with logistic activation function, adaptive learning rate, each layer has 100 hidden nodes, and 1000 maximum iterations.

## 4.2 Naïve Bayes

Naïve Bayes is the baseline for classification model, which should be used for comparison with other models. Naïve Bayes is a model based on statistical assumption that all features in the given dataset are independent to each other. Considering the previous discussion of the correlation coefficient between each pair of features, this assumption may have some effect on the prediction results. The Naïve Bayes model can compute at a very fast manner compared to other classification models, and its performance is usually very well with high dimensional data since its assumption is based on the probability theory hypothesis. Though Naäive Bayes classifier usually can generate good predictions, it has an overfitting issue that has to be dealt with in a careful manner.

## 4.3 Support Vector Machine

Compared to other linear classification models, support vector machine classifier performs very well in both linear and non-linear boundary. With the help of different kernels like the radial basis function kernel, it can have a superior performance on high dimensional dataset. It also can deal with outliers since the classification only depends on support vectors. However, it usually requires a long computation time on large dataset, and selection of the right kernel is not simple. In this report, the SVM model is constructed with rbf kernel and penalty parameter C equals to 1.

## 4.4 K-Nearest Neighbor

As a lazy classifier that has no training step, K-NN is a robust supervised model that is suitable for many situations. Since there is no assumptions on the attributes’ dependencies as well as the setting of parameters, the K-NN is easy to implement for both classification and regression problems. With no training in building K-NN models, every incoming testing tuples has to come up with an entire search of closest neighbors that loop through the whole dataset. This means that the computation efficiency of K-NN models declines following the increases of the size of dataset and the model parameters are hidden inside. In addition, K-NN model is extremely sensitive to outliers; therefore, the dataset preprocessing procedure is essential with applying the K-NN model to a dataset before its training started. In this report, the K-NN model is constructed with 5 neighbors.

## 4.3 Decision Tree

Unlike other classification models with complex model coefficients and hidden implementation that humans cannot easily interpret, decision tree vividly express the decision procedure of a given dataset by mimicking human decision making. The result usually has well separated decision boundaries that processes both numerical and categorical attributes. The limitation of decision tree is also clear that without using bagging or boosting technologies, it is easy to overfit and pruning is required to generalize the model.

# 5 Evaluation and Validation

After the implementation of the above models with testing different parameters with corresponding dataset preprocessing methods, there are many results produced under different setups need to be compared. In order to get a comprehensive comparison of the results generated by different models, efficiency evaluation and validation methods are used in this report including cross validation, confusion matrix, and receiver operating characteristic curve.

## 5.1 Result Evaluation

Table 1 shows the comparison of accuracy among three missing value handling methods. The three methods are dropping the “unknown” blocks, using “unknown” blocks as a class and predict them as missing values by applying Decision Tree or Naïve Bayes. Both applying “unknown” as class method and prediction replacement method obtain better accuracy compared to directly drop tuples with “unknown” blocks.

TABLE 1  
Accuracy Comparison

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Algorithm | MLPNN | NB | SVM | KNN | DT |
| Dropping | 0.87 | 0.70 | 0.88 | 0.87 | 0.82 |
| “unknown” as a Class | 0.89 | 0.84 | 0.89 | 0.89 | 0.83 |
| Predicting Imputation | 0.89 | 0.87 | 0.89 | 0.85 | 0.83 |
| Dropping  (PCA = 5) | 0.87 | 0.87 | 0.87 | 0.86 | 0.77 |
| “unknown”  as a Class  (PCA = 5) | 0.88 | 0.88 | 0.88 | 0.88 | 0.79 |
| Predicting Imputation  (PCA = 5) | 0.88 | 0.88 | 0.88 | 0.87 | 0.78 |

*Table 1 has 5 different models accuracy with different data preprocessing methods.*

For better illustration of models’ performance without using PCA, a line-scatter plot is plotted (Fig. 10). Based on this plot, using 'unknown' as a class to replace missing value can get the highest result; on the other hand, using predicting imputation method has the lowest deviation between each model. In the end, directly drop missing value as the baseline has the lowest accuracy and the largest deviation.

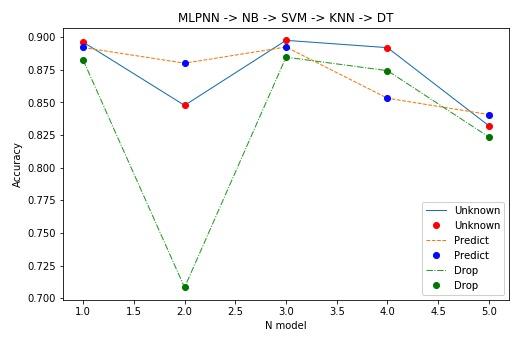


Fig. 10 Accuracy line-scatter plot of models without PCA.

For better illustration of model’s performance using PCA, another line-scatter plot is plotted (Fig. 11). Compared to the previous plot, PCA generates a huge difference for each preprocessing method. First, PCA does not affect model accuracy. Second, every model has a similar deviation. The reason of that is only the top main 5 components are kept for training, and each attribute has a normal distribution. This plot also proves using appropriate missing value replacement policy can obtain better results.

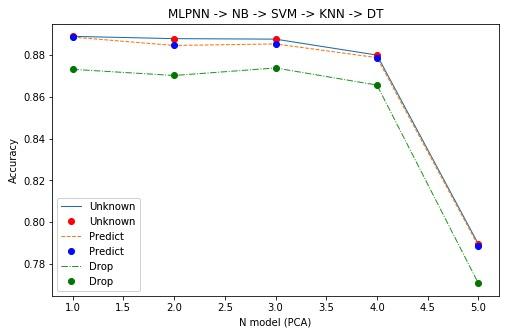


Fig. 11 Accuracy line-scatter plot of models with PCA.

Table 2 shows comparison of computation time of each data preprocessing method against using PCA or not. Each column calculates the summation of training time with all 5 kinds of model. The result indicates that with PCA preprocessing training time can be significantly reduced.

TABLE 2  
Training Time Comparison

|  |  |  |  |
| --- | --- | --- | --- |
| Time (s) | Drop | Unknown | Prediction |
| No PCA | 70 | 123 | 102 |
| PCA = 5 | 50 | 60 | 69 |

*Table 2 has 2 kinds of training time with different data preprocessing methods.*

## 5.2 Model Validation

The receiver operating characteristic curve is a probability curve reflecting the sensitivity and specificity of a given classification model. In the ROC and AUC plot, false positive rate (FPR) is on the x-axis, and true positive rate (TPR) is on the y-axis. Each point of the ROC curve reflects the sensitivity to the same signal simulus. By comparing the ROC curve for different models, the performance of each classifier can be compared under different threshold. The Area Under the Curve (AUC) value represent the general performance of the model. When the AUC value equals to 1, the model is a perfect classifier that can generate perfect prediction.

It’s generally not possible to have a perfect classifier while most of the classifier has an AUC value between 0.5 and 1. Those models with AUC equals to 0.5 are meaningless since they are basically random guessing. Below plotted five classification models’ ROC and AUC graph for model validation analysis.

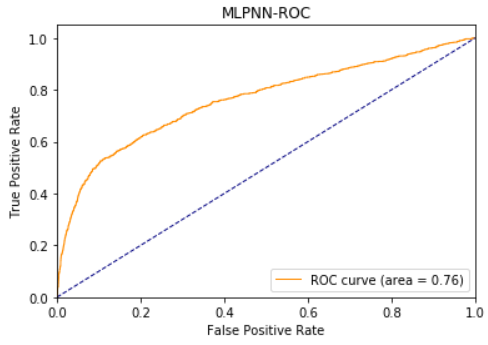


Fig. 12 ROC & AUC of Neural Network Model

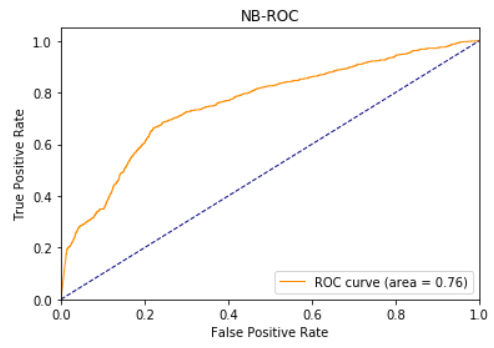


Fig. 13 ROC & AUC of Naïve Bayes Model

Fig. 12 and Fig. 13 shows the ROC curve and AUC of MLPNN and Näive Bayes models. It is worth noting that the AUC of two model is the same meaning that the MLPNN model owns the similar accuracy compared to the Naïve Bayse model. Besides the similarity of AUC, the performance of these two classifiers are not quite the same according to the ROC curve. The curve of MLPNN is smoother compared to the Naïve Bayes model, which means that the MLPNN model can have a stable result with higher specificity.

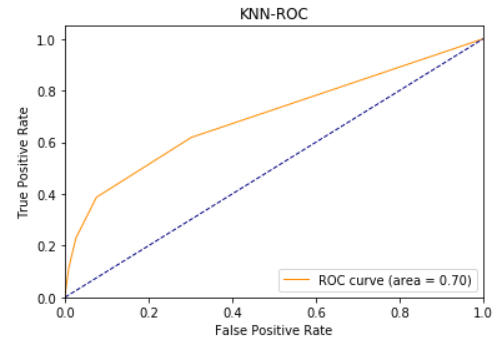


Fig. 14 ROC & AUC of K-Nearest Neighbor Model

The performance of KNN is shown by the ROC curve in Fig. 14. Comparing to the ROC curve of the MLPNN model and NB model, the AUC value is smaller that indicates an unsatisfactory prediction result. The actual performance of the KNN is not inferior considering that the ROC curve of KNN is smoother than the curve of Naïve bayes. Therefore, KNN model has a better performance with higher specificity.

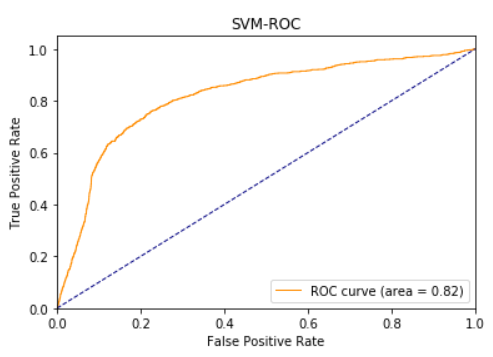
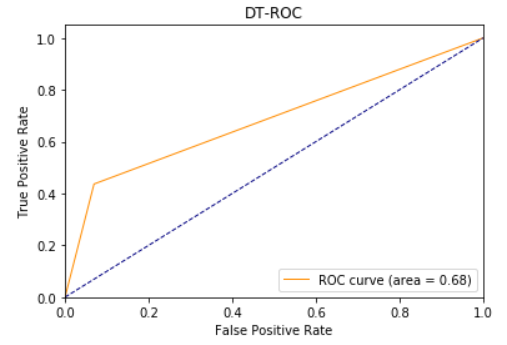


Fig. 15 ROC & AUC of Support Vector Machine Model

As one of the best classifiers for the linear separable data, SVM model generates the best result with the testing data under all threshold. As shown in Fig. 15, the AUC value of SVM model is 0.82 which beats all other models and the ROC curve is very close to the perfect model.



# Fig. 16 ROC & AUC of Decision Tree

Different from other models, the decision tree classifier generates the worst prediction result by looking at Fig.16. The AUC value is low at 0.68, which is close to random prediction. The ROC curve is steep and sharp indicates an unstable performance.

# 6 Conclusion

This report is an extension of the significant paper *An Efficient CRM-Data Mining Framework for the Prediction of Customer Behavior* that proposed a framework that can process CRM data to give insights to the customer’s behavior and make predictions on their future decisions. Different data visualization techniques are used in this report to have a general idea of the dataset. Data preprocessing is conducted by three different approaches to deal with missing values, and PCA is used to reduce feature dimension. The dataset is then fed to five different machine learning models to find out their performances. By using prediction accuracy, computation time and ROC-AUC as metrics, MLPNN, NB and SVM with PCA and missing value imputation are proved to provide the best performing results.

**References**

1. T. Femina Bahari, “An Efficient CRM-Data Mining Framework for the Prediction of Customer Behaviour,” Procedia Computer Science*,* pp. 725-731, 2015.
2. Seyed M. Seyed, “Cluster Analysis Using Data Mining Approach to Develop CRM Methodology to Access the Customer Loyalty,” Experts Systems with Applications, pp. 5259-5264, 2010.
3. E.W.T. Ngaia, “Application of data mining techniques in customer relationship management: A literature review and classification,” Experts Systems with Applications, pp. 2592-2602, 2009.
4. C.C. Luk, “Design of an Intelligent Customer Identification Model in eCommerce Logistics Industry,” [MATEC Web of Conferences](https://www.researchgate.net/journal/2261-236X_MATEC_Web_of_Conferences), col. 255(6), 2019.
5. Ibrahim Mousa AI-Zuabi, “Predicting customer’s gender and age depending on mobile phone data,” 2019.

# Appendix

## Github Link

<https://github.com/pichao314/CRM>

## Source Code

import pandas as pd

import seaborn as sns

import numpy as np

from matplotlib import pyplot as plt

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import cross\_val\_score, train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.tree import DecisionTreeClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier, plot\_tree, export\_graphviz

from sklearn.svm import SVC

from sklearn.neural\_network import MLPClassifier

import time

from sklearn.metrics import classification\_report, confusion\_matrix

from sklearn.metrics import accuracy\_score

from sklearn.metrics import roc\_curve, auc

'''

Read the data

'''

#read the data and drop useless features

data = pd.read\_csv('original\_data.csv')

data = data.drop('Unnamed: 0', axis = 1)

#encode the target value

le = LabelEncoder()

data['y']= le.fit\_transform(data['y'])

'''

Find the outlier of the dataset and figure out the boundary

'''

#plot the box plot

f1 = sns.boxplot(data['duration'])

f1.figure.set\_size\_inches(8,5)

plt.show()

#figure out the outlier boundary

q3 = np.percentile(data['duration'], [75])

q1 = np.percentile(data['duration'], [25])

iqr = q3 - q1

print("Upper\_bound:", q3 + 1.5\*iqr)

print("Lower\_bound:", q1 - 1.5\*iqr)

#plot of feature distribution

sns.pairplot(data.iloc[:,:15])

'''

The correlation map

'''

corr = data.corr()

plt.figure(figsize=(15, 15))

sns.heatmap(corr, square=True, linewidths=.5, annot=True, fmt='.2f', cmap='coolwarm')

'''

Check and handle missing values

'''

#loop the data to count the number of missing values, which treated as unknown

missing\_dict = {}

for col in data.columns:

missing\_dict[col] = 0

for item in data[col]:

if item == 'unknown':

missing\_dict[col] += 1

#drop the useless column

udata = data.drop(['Unnamed: 0', 'duration', 'cons.price.idx', 'euribor3m', 'nr.employed'], axis = 1)

# encoding the categorical data,

le1 = LabelEncoder()

for label in udata.columns:

udata[label]= le1.fit\_transform(udata[label])

# scaling the numerical data

for col in ['campaign', 'pdays', 'emp.var.rate', 'cons.conf.idx']:

sc = StandardScaler(with\_mean=True, with\_std=True)

udata[col] = sc.fit\_transform(udata[col].values.reshape(-1,1))

# separate the data and target value

ux = udata.drop('y', axis = 1)

uy = udata['y']

# split the data into training and testing data by ratio 0.2

uX\_train, uX\_test, uY\_train, uY\_test = train\_test\_split(ux, uy, test\_size = 0.2)

'''

Using decision tree or Naive Bayes to impute missing value

'''

#build and fit the decision tree

DT = DecisionTreeClassifier()

DT.fit(X\_train, Y\_train)

#build and fit the naive bayes

NB = GaussianNB()

NB.fit(X\_train, Y\_train)

#predict the result

Y\_pred1 = NB.predict(X\_test)

Y\_pred2 = DT.predict(X\_test)

#print the accuracy

print('accuracy of DT:', accuracy\_score(Y\_test, Y\_pred2))

print('accuracy of NB:', accuracy\_score(Y\_test, Y\_pred1))

#collect the missing value

marital\_missing = []

marital\_index = []

for i, row in pdata.iterrows():

if row[2] == 3:

marital\_missing.append(row)

marital\_index.append(i)

#drop the row with missing values

marital\_data = pdata.drop(marital\_index)

#rebuilt the data set for predicting missing values

marital\_test = pd.DataFrame(marital\_missing, columns = col)

marital\_test = marital\_test.drop('marital', axis = 1)

marital\_x = marital\_data.drop('marital', axis = 1)

marital\_y = marital\_data['marital']

#split the new data set

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(marital\_x, marital\_y, test\_size = 0.2)

#impute the missing value by decision treeand naive bayes

DT = DecisionTreeClassifier()

DT.fit(X\_train, Y\_train)

NB = GaussianNB()

NB.fit(X\_train, Y\_train)

Y\_pred1 = NB.predict(X\_test)

Y\_pred2 = DT.predict(X\_test)

#print the score after imputing

print('accuracy of DT:', accuracy\_score(Y\_test, Y\_pred2))

print('accuracy of NB:', accuracy\_score(Y\_test, Y\_pred1))

#predict the marital missing value

NB = GaussianNB()

NB.fit(marital\_x, marital\_y)

Y\_pred3 = NB.predict(marital\_test)

marital\_test['marital'] = Y\_pred3

pdata = pd.concat([marital\_data, marital\_test])

'''

Drop the missing value

'''

#get the index of missing value

missing\_list = []

for i in range(len(mdata.index)):

if 'unknown' in mdata.iloc[i].values:

missing\_list.append(i)

#drop the columns

mdata = mdata.drop(missing\_list)

mdata = mdata.drop(['Unnamed: 0','duration','cons.price.idx', 'euribor3m', 'nr.employed', 'poutcome'], axis = 1)

'''

Accuracy of unknown policy without using PCA

'''

start\_time1 = time.time()

#the MLPNN Classifier

uml = MLPClassifier(activation = 'logistic', learning\_rate = 'adaptive', max\_iter = 1000, verbose = False)

uml.fit(uX\_train, uY\_train)

uy1 = uml.predict(uX\_test)

au1 = accuracy\_score(uY\_test, uy1)

print('accuracy of MLPNN:', au1)

#the Naive Bayes Classifier

unb = GaussianNB()

unb.fit(uX\_train, uY\_train)

uy2 = unb.predict(uX\_test)

au2 = accuracy\_score(uY\_test, uy2)

print('accuracy of NB:', au2)

#the SVM Classifier

usvm = SVC()

usvm.fit(uX\_train, uY\_train)

uy3 = usvm.predict(uX\_test)

au3 = accuracy\_score(uY\_test, uy3)

print('accuracy of SVM:',au3)

#the KNN Classifier

uknn = KNeighborsClassifier(n\_neighbors=5)

uknn.fit(uX\_train, uY\_train)

uy4 = uknn.predict(uX\_test)

au4 = accuracy\_score(uY\_test, uy4)

print('accuracy of KNN:', au4)

#the Decision Tree Classifier

udt = DecisionTreeClassifier()

udt.fit(uX\_train, uY\_train)

uy5 = udt.predict(uX\_test)

au5 = accuracy\_score(uY\_test, uy5)

print('accuracy of DT:', au5)

ut = time.time() - start\_time1

'''

Accuracy of prediction policy without using PCA

'''

start\_time2 = time.time()

#the MLPNN Classifier

pml = MLPClassifier(activation = 'logistic', learning\_rate = 'adaptive', max\_iter = 1000, verbose = False)

pml.fit(pX\_train, pY\_train)

py1 = pml.predict(pX\_test)

ap1 = accuracy\_score(pY\_test, py1)

print('accuracy of MLPNN:', ap1)

#the Naive Bayes Classifier

pnb = GaussianNB()

pnb.fit(pX\_train, pY\_train)

py2 = pnb.predict(pX\_test)

ap2 = accuracy\_score(pY\_test, py2)

print('accuracy of NB:', ap2)

#the SVM Classifier

psvm = SVC()

psvm.fit(pX\_train, pY\_train)

py3 = psvm.predict(pX\_test)

ap3 = accuracy\_score(pY\_test, py3)

print('accuracy of SVM:',ap3)

#the KNN Classifier

pknn = KNeighborsClassifier(n\_neighbors=5)

pknn.fit(pX\_train, pY\_train)

py4 = uknn.predict(pX\_test)

ap4 = accuracy\_score(pY\_test, py4)

print('accuracy of KNN:', ap4)

#the Decision Tree Classifier

pdt = DecisionTreeClassifier()

pdt.fit(pX\_train, pY\_train)

py5 = pdt.predict(pX\_test)

ap5 = accuracy\_score(pY\_test, py5)

print('accuracy of DT:', ap5)

pt = time.time() - start\_time2

'''

Accuracy of drop policy without using PCA

'''

start\_time3 = time.time()

#the MLPNN Classifier

mml = MLPClassifier(activation = 'logistic', learning\_rate = 'adaptive', max\_iter = 1000, verbose = False)

mml.fit(mX\_train, mY\_train)

my1 = mml.predict(mX\_test)

am1 = accuracy\_score(mY\_test, my1)

print('accuracy of MLPNN:', am1)

#the Naive Bayes Classifier

mnb = GaussianNB()

mnb.fit(mX\_train, mY\_train)

my2 = mnb.predict(mX\_test)

am2 = accuracy\_score(mY\_test, my2)

print('accuracy of NB:', am2)

#the SVM Classifier

msvm = SVC()

msvm.fit(mX\_train, mY\_train)

my3 = msvm.predict(mX\_test)

am3 = accuracy\_score(mY\_test, my3)

print('accuracy of SVM:',am3)

#the KNN Classifier

mknn = KNeighborsClassifier(n\_neighbors=5)

mknn.fit(mX\_train, mY\_train)

my4 = mknn.predict(mX\_test)

am4 = accuracy\_score(mY\_test, my4)

print('accuracy of KNN:', am4)

#the Decision Tree Classifier

mdt = DecisionTreeClassifier()

mdt.fit(mX\_train, mY\_train)

my5 = mdt.predict(mX\_test)

am5 = accuracy\_score(mY\_test, my5)

print('accuracy of DT:', am5)

mt = time.time() - start\_time3

'''

Accuracy plot of three policy without using PCA

'''

x = [1, 2, 3 ,4 ,5 ]

y1 = [au1, au2, au3, au4, au5]

y2 = [ap1, ap2, ap3, ap4, ap5]

y3 = [am1, am2, am3, am4, am5]

#plot the figure of three policy

plt.figure(figsize = (8, 5))

plt.plot(x,y1,"-",x,y1,"ro",linewidth=1, label = 'Unknown')

plt.plot(x,y2,"--", x,y2,"bo",linewidth=1, label = 'Predict')

plt.plot(x,y3,"-.",x,y3,"go",linewidth=1, label = 'Drop')

plt.xlabel("N model")

plt.ylabel("Accuracy")

plt.title("MLPNN -> NB -> SVM -> KNN -> DT")

plt.legend()

plt.show()

#From the comparison result, we can see using 'unknown' to replace missing value can get the highest result, however, using prediction method, it has the lowest deviation between each model finally, directly dropping missing value cause the lowest accuracy

'''

Accuracy plot of three policy with using PCA

'''

import matplotlib.pyplot as plt

x = [1, 2, 3 ,4 ,5 ]

y11 = [au11, au21, au31, au41, au51]

y21 = [ap11, ap21, ap31, ap41, ap51]

y31 = [am11, am21, am31, am41, am51]

plt.figure(figsize = (8, 5))

plt.plot(x,y11,"-",x,y11,"ro",linewidth=1, label = 'Unknown')

plt.plot(x,y21,"--", x,y21,"bo",linewidth=1, label = 'Predict')

plt.plot(x,y31,"-.",x,y31,"go",linewidth=1, label = 'Drop')

plt.xlabel("N model (PCA)")

plt.ylabel("Accuracy")

plt.title("MLPNN -> NB -> SVM -> KNN -> DT")

plt.legend()

plt.show()

'''

PCA comparison

'''

#plot the unknown policy comparison

x1 = [1, 2, 3, 4, 5]

uwp = [au1, au2, au3, au4, au5]

up = [au11, au21, au31, au41, au51]

plt.figure(figsize = (7, 5))

plt.plot(x1,uwp,"-",x1,uwp,"ro",linewidth=1, label = 'without PCA')

plt.plot(x1,up,"--", x1,up,"bo",linewidth=1, label = 'PCA')

plt.xlabel("N model (unknown policy)")

plt.ylabel("Accuracy")

plt.title("MLPNN -> NB -> SVM -> KNN -> DT")

plt.legend()

plt.show()

#plot the prediciton policy comparison

x2 = [1, 2, 3, 4, 5]

pwp = [ap1, ap2, ap3, ap4, ap5]

pp = [ap11, ap21, ap31, ap41, ap51]

plt.figure(figsize = (7, 5))

plt.plot(x2,pwp,"-",x2,pwp,"ro",linewidth=1, label = 'without PCA')

plt.plot(x2,pp,"--", x2,pp,"bo",linewidth=1, label = 'PCA')

plt.xlabel("N model (Prediction policy)")

plt.ylabel("Accuracy")

plt.title("MLPNN -> NB -> SVM -> KNN -> DT")

plt.legend()

plt.show()

#plot the drop policy comparison

x3 = [1, 2, 3, 4, 5]

mwp = [am1, am2, am3, am4, am5]

mp = [am11, am21, am31, am41, am51]

plt.figure(figsize = (7, 5))

plt.plot(x3,mwp,"-",x3,mwp,"ro",linewidth=1, label = 'without PCA')

plt.plot(x3,mp,"--", x3,mp,"bo",linewidth=1, label = 'PCA')

plt.xlabel("N model (Drop policy)")

plt.ylabel("Accuracy")

plt.title("MLPNN -> NB -> SVM -> KNN -> DT")

plt.legend()

plt.show()

'''

ROC

'''

ML1\_score = uml.predict\_proba(uX\_test)

NB1\_score = unb.predict\_proba(uX\_test)

SVM1\_score = usvm.decision\_function(uX\_test)

KNN1\_score = uknn.predict\_proba(uX\_test)

DT1\_score = udt.predict\_proba(uX\_test)

fpr = dict()

tpr = dict()

roc\_auc = dict()

for i in range(2):

fpr[i], tpr[i], \_ = roc\_curve(uY\_test, ML1\_score[:, i])

roc\_auc[i] = auc(fpr[i], tpr[i])

lw = 1

plt.plot(fpr[1], tpr[1], color='darkorange',

lw=lw, label='ROC curve (area = %0.2f)' % roc\_auc[1])

plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('MLPNN-ROC')

plt.legend(loc="lower right")

plt.show()

fpr = dict()

tpr = dict()

roc\_auc = dict()

for i in range(2):

fpr[i], tpr[i], \_ = roc\_curve(uY\_test, NB1\_score[:, i])

roc\_auc[i] = auc(fpr[i], tpr[i])

lw = 1

plt.plot(fpr[1], tpr[1], color='darkorange',

lw=lw, label='ROC curve (area = %0.2f)' % roc\_auc[1])

plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('NB-ROC')

plt.legend(loc="lower right")

plt.show()

fpr = dict()

tpr = dict()

roc\_auc = dict()

for i in range(2):

fpr[i], tpr[i], \_ = roc\_curve(uY\_test, SVM1\_score)

roc\_auc[i] = auc(fpr[i], tpr[i])

lw = 1

plt.plot(fpr[1], tpr[1], color='darkorange',

lw=lw, label='ROC curve (area = %0.2f)' % roc\_auc[1])

plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('SVM-ROC')

plt.legend(loc="lower right")

plt.show()

fpr = dict()

tpr = dict()

roc\_auc = dict()

for i in range(2):

fpr[i], tpr[i], \_ = roc\_curve(uY\_test, KNN1\_score[:, i])

roc\_auc[i] = auc(fpr[i], tpr[i])

lw = 1

plt.plot(fpr[1], tpr[1], color='darkorange',

lw=lw, label='ROC curve (area = %0.2f)' % roc\_auc[1])

plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('KNN-ROC')

plt.legend(loc="lower right")

plt.show()

fpr = dict()

tpr = dict()

roc\_auc = dict()

for i in range(2):

fpr[i], tpr[i], \_ = roc\_curve(uY\_test, DT1\_score[:, i])

roc\_auc[i] = auc(fpr[i], tpr[i])

lw = 1

plt.plot(fpr[1], tpr[1], color='darkorange',

lw=lw, label='ROC curve (area = %0.2f)' % roc\_auc[1])

plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('DT-ROC')

plt.legend(loc="lower right")

plt.show()