

Group 3:

Giai Tran

Y Nguyen

Thuy Nguyen

Syed

Instructor: Chistoph F. Eick

TA: Nour Smaouri

University of Houston

2019

DATA SCIENCE

REPORT

**INDEX**

1

**Introduction**

[*Page 3*](#_INTRODUCTION)

2

**Preprocessing data**

[*Page 4*](#_PREPROCESSING)

3

**Experimental Results**

[*Support Vector Machine – page 7*](#_Support_Vector_Machine)

[*K-Neighbor Nearest ---------page 10*](#_K-Neighbor_Nearest_Approach)

[*Neural Network---------------page 13*](#_Neural_Network)

[*Random Forest---------------page 14*](#_Random_Forest_Approach)

4

**Comparison & Conclusion**

[*Page 15*](#_Comparison_&_Conclusion)

# INTRODUCTION

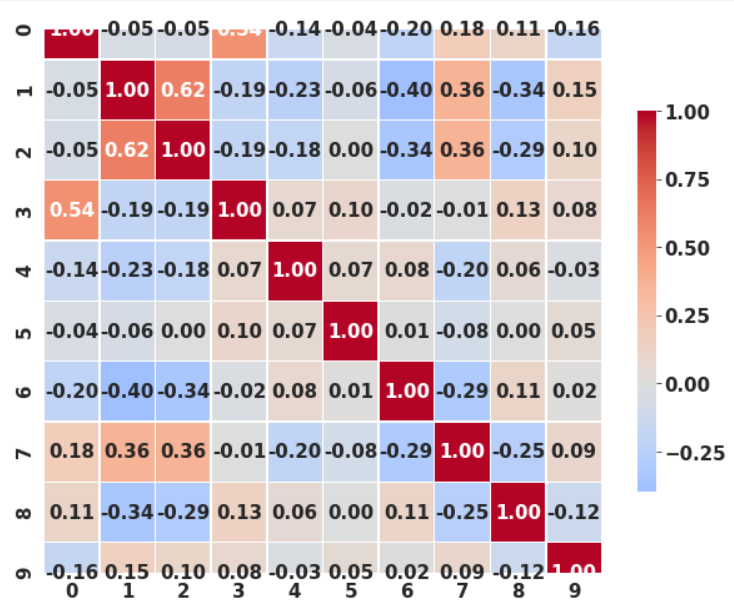
We are given a dataset named Contraceptive Method Choice. Our mission is performing classification approaches to better understand our dataset. In order to archive that, we apply four of many methods that were discussed in class lecture. They are Neural Network, Support Vector Machine (SVM), K Nearest Neighbors (KNN), and Random Forest.

We are given 10 basic explanatory variables, including wife’s age, wife’s education, husband’s education, number of children ever born, wife’s religion, wife’s now working, husband’s occupation, standard of living index, media exposure, and contraceptive method used. Our dataset contains 1473 instances without any null value. This is a multiclass classification problem.

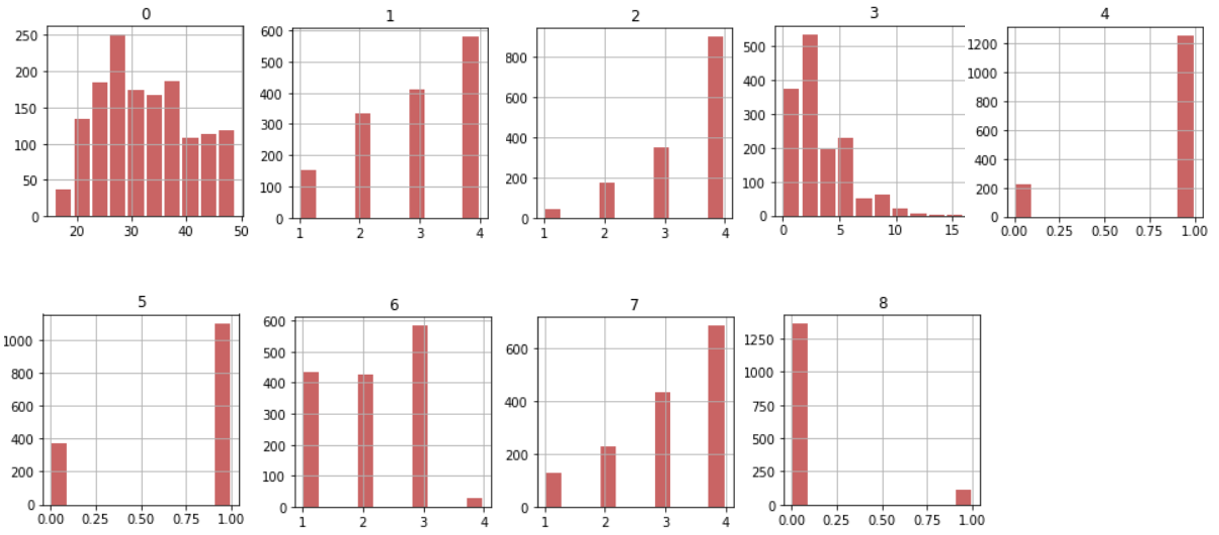
We have to develop methods that are able to classify each instance based on the given dataset. Our project is written on Python language and uses scikit-learn as our primary tool. All of our group member are working on Google Colab environment. Google Colab is a fast, online cloud service provided by Google for education purposes, and there is no installation needed.

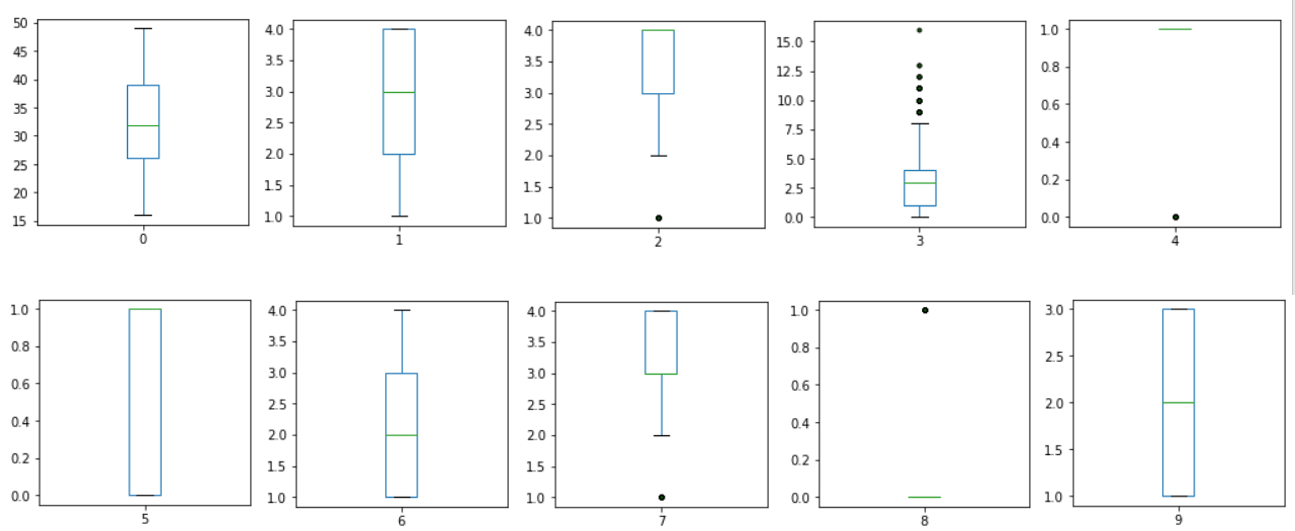
# PREPROCESSING

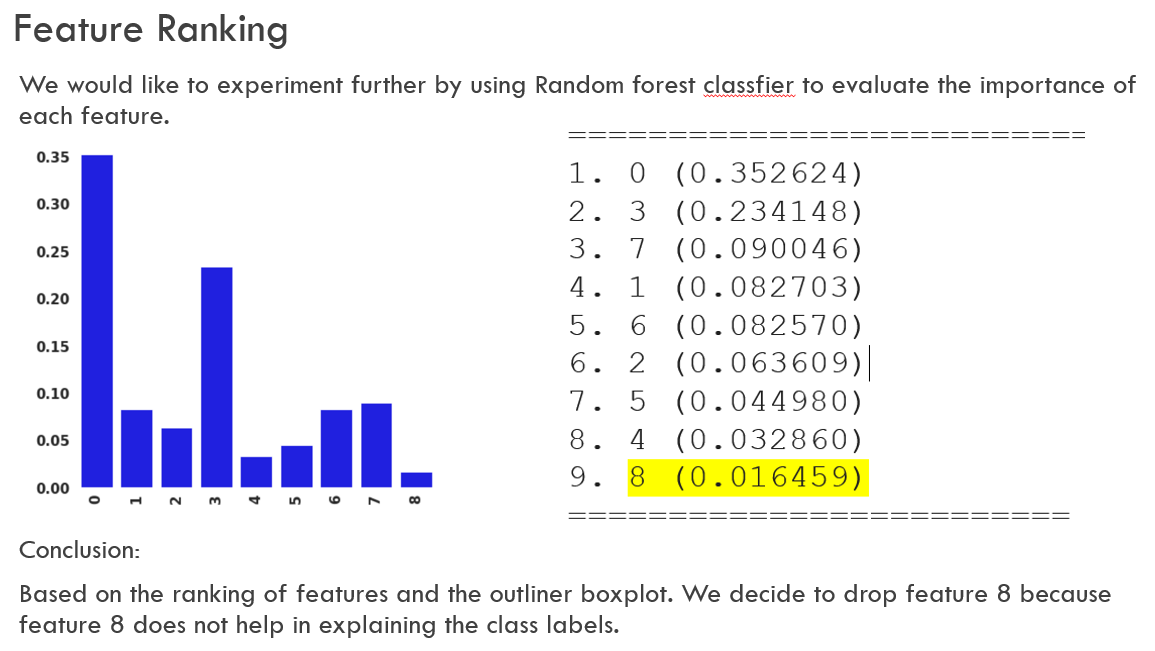
First, we check high correlation of features because there is a high chance that a high correlation feature impacts the performance of the model. Feature correlation is a good way to better understand the relationship between multiple variables and attributes in our dataset.



By checking high correlation features, we feature out which feature contributes the most in our dataset. For those features with high correlation, we may drop them from our dataset. In the other hand, for those one with low correlation, we can try to combine them to create a new feature.

Second important process is identifying Noise in our dataset. We must eliminate unwanted data which don’t help in explaining the itself, or the relationship between feature and class label. Removing Noise enables our model to train faster, improve the accuracy and reduce overfitting. We visualize the features distribution and outliers using histogram and boxplot. In addition, we perform feature ranking by using Random Forest Classifier.

We need to go to check the outlier of data before use. Because the noise of data can affect deeply to the final performance of model. We plot the outlier of each feature in the figures as follows.



Observing from the graphs, we can identify Noise in dataset and eliminate them. We have done 75% work so far, the next step needed is handling the normalization. As the descriptive statistic of dataset shows that there are many differences in range of values in each feature, especially for the first feature. Rescaling is the simplest method and consists in rescaling the range of features to the range in (0, 1). We try serveral methods such as Min-Max scaling, Normalizer, and finally end up with Standard Scaler due to better performance in testing.

We observe the number of instances in each class and figure out class 2 is a minority class comparing to class 1 and class 3. To deal with it, we decide to use oversampling method by applying SMOTE algorithm. Basically, SMOTE algorithm creates synthetic (not duplicates) samples of the minority class and applies KNN approach where it selects K nearest neighbors, joins them and creates the synthetic samples in the space.

# Support Vector Machine Approach

From the above preprocessing data progress, we found that data may be imbalanced. Hence, we will split into two cases: data set is balance, and the data set is imbalanced. We used SMOTE to handle imbalanced data set.

Support Vector Machine applied Cross Validation will be mentioned as SVM\_CV in this report.

Support Vector Machine applied Cross Validation and applied SMOTE will be mentioned as SVM\_CV\_SMOTE.

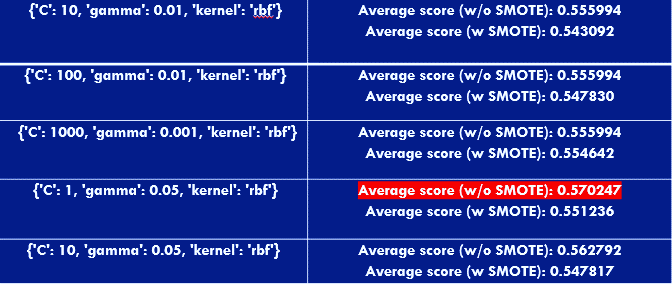
We applied cross validation by randomly splitting our cleaned dataset into 10 consecutive folds using K-Fold cross validation. This method provides train/test indices to split data in train/test sets.

Firstly, we run the function SVC(support vector classifier) with the default parameter values for both SMV\_CV and SVM\_CV\_SMOTE. The outcome results are as follow. The result shows that applying SMOTE give us a lower accuracy scores in classifying our data. Therefore, applying SMOTE won’t be helpful.

* **Improve the accuracy result by GridSearchCV method**

In order to improve the accuracy of our SVC method, we apply parameter tuning. We used GridSearchCV(grid search cross validation method) in scikit-learn library to estimate the best parameter sets that may be able to help our SVC produces a better result.

* **Create Parameter Candidates:** Before looking for which combination of parameter values produces the most accurate model, we must specify the different candidate values we want to try.
  + C: 0.1, 1, 10, 100, 1000
  + gamma: 0.0001,0.001,0.05,0.01,0.5,1
  + kernels: linear, RBF
* **Find the best parameter sets:** The grid search will try all combinations of parameter values and select the set of parameters which provides the most accurate model for each fold.
* **Result:** After running cross-validation procedures, search the best parameters for SVM algorithm and recording a list of the best parameter sets, we went back and applied these parameters to your SVC method. According to the grid search result, all the accuracy scores from using linear kernel are lower than RBF kernel.



By using parameter tuning, we found out the best fit parameter for our SVC. As the result, we improved the average accuracy for our SVM\_CV from 55.60% to 57.02% which is about 1.5%.

# K-Neighbor Nearest Approach

* **Training and Predictions**

It is extremely straight forward to train the KNN algorithm and make predictions with it, especially when using Scikit-Learn. The first step is to import the KNeighborsClassifier class from the sklearn.neighbors library.

* **Choosing parameters:**
* ***K – Values***

In KNN, finding the value of k which is optimal is not easy. A small value of k means that noise will have a higher influence on the result and a large value make it computationally expensive. There is no ideal value for K and it is selected after testing and evaluation, however to start out, 5 seems to be the most commonly used value for KNN algorithm.

In fact, data scientists usually choose as an odd number if the number of classes is 2. Another simple approach to select k is set k = sqrt(n). where n = number of data points in training data.

In our case, the number of classes is 3, and we decided to go with k = 5, 7, and 37. Where 37 = square of (1326). 1326 is number of data points in training data All these values cannot divide by 3. So we can avoid the cases that the number of objects belong to each class are equal.

* ***Leaf – size***

This affects the speed of construction and query. In general, the larger leaf-size, the closer neighbors the algorithm picks . This is because the tree's main purpose is to reduce the number of candidates for the neighbors. The default for leaf-size is 30. However, we tried it out, and with our data set, the leaf\_size gave us best result at value of 10.

* ***Algorithm = ‘auto’:***

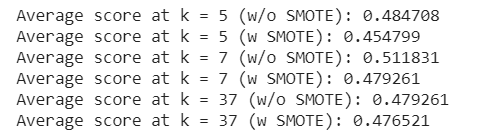
We choose auto, which is the default of this parameter. ‘auto’ will attempt to decide the most appropriate algorithm based on the values passed to fit method.

* **Train Test Split**

To avoid over-fitting, we will divide our dataset into training and test splits, which gives us a better idea as to how our algorithm performed during the testing phase.

We using K-Fold validation. With k = 10, we split dataset into 10 consecutive folds. The dataset is split into 90% train data and 10% test data. This means that out of total 1474 records, the training set will contain 1326 records and the test set contains 147 of those records.

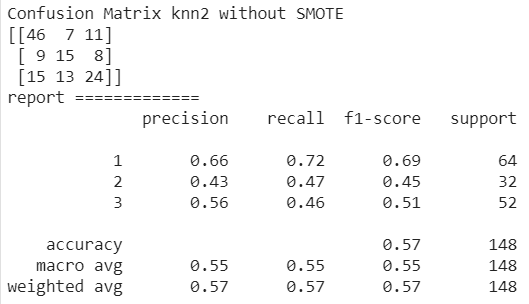
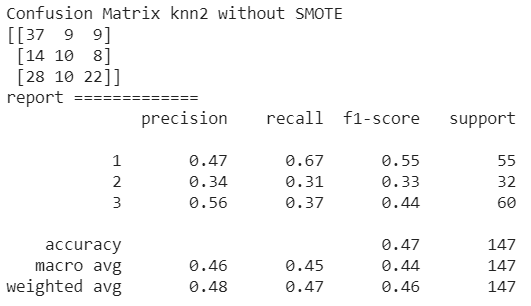
By using 10-fold validation, the average ACCURACY is about ~50% for all test cases



Our KNN algorithm get highest average score at K=7 (without using SMOTE)

* **Evaluating the KNN Algorithm**

For evaluating an algorithm, confusion matrix, precision, recall and f1 score are the most commonly used metrics.



*The Confusion\_Matrices and Classification\_reports of the winner.*

The results of our best show that our KNN algorithm was able to classify all the 147 records in the test set with 57% accuracy at most and 47% at lowest, average is 51.18%, which is not really excellent, but not also bad. Although the algorithm performed not badly with this dataset, don't expect the same results with all applications. As noted earlier, KNN doesn't always perform as well with high-dimensionality or categorical features.

# Neural Network

Neural Networks are a set of algorithms that function like a human brain. NN can design, recognize patterns and help solve regression or classification problems. First, we split the dataset into ten subsets using 10-fold cross validation and train them. The first two results use the default configuration of and the two last use Grid-search to perform parameter tuning. As we observe results, the last two perform much better after tuning parameters using Gird-search. Last but not least, experiment applied SMOTE algorithm does not improve the accuracy.

# Random Forest Approach

By applying cross validation with k-fold =10, we recorded our accuracy scores after each round and calculate the average accuracy scores at the end.

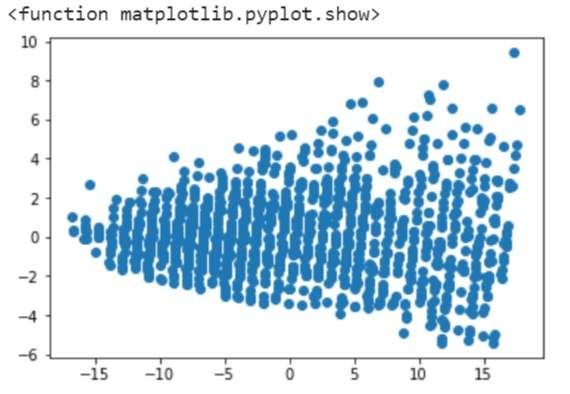
We did the same procedure for the dataset with applied SMOTE method.

As the result, the accuracy scores are about 51.5% to 53%. The accuracy scores don’t vary much. Interestingly, we found out that sometimes the accuracy score with applied SMOTE is higher than the accuracy score without SMOTE.

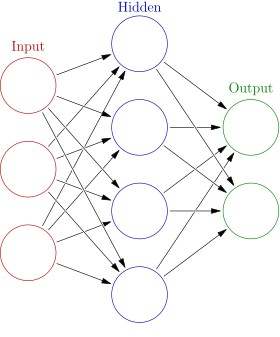
# Comparison & Conclusion Approach

* **Comparison:**

Among of four classifier methods, Neural Network outperform the others. Let’s discuss how other methods perform first. KNN is one of the simplest of classification algorithms for supervise learning. The idea of the algorithm is to search for the closest match of the test data in feature space. Support Vector Machine (SVM) algorithm builds a model that assigns new examples to one category or the other. Both KNN and SVM algorithms are work based on distances.



We use Principal component analysis (PCA) method to transfer the dataset into a set of values of linearly uncorrelated variables. The scatter-plot shows classes overlay each other and that result in reducing efficient of KNN and SVM algorithms. Besides that, Random Forest (RF) algorithm constructs a multitude of decision trees using GINI index. However, GINI index treats each feature independently. Moreover, features in our dataset have a weak relationship comparing to class label (check heatmap above). It is the main reason causing RF algorithm does not perform well in this scenario. Last, we are going to discuss Neural Network (NN) algorithm which is the best classifier of our dataset.

In NN, each layer of nodes trains on a distinct set of features based on the previous layer’s output. The further we advance into the neural net, the more complex the features our nodes can recognize, since they aggregate and recombine features from the previous layer. The main point is that NN algorithm tries to combine features and calculate their weight in several ways. NN algorithm outperforms others due to the weak relationship among features. We will discuss about combining features more detail later in the report.

* **Conclusion:**

In conclusion, most features in dataset have weak correlation comparing to each other and class labels. There is no important feature help better understand the class labels. Features overlaying significantly decrease efficient of KNN, SVM performance. Imbalance among three classes isn’t high so the accuracy without using SMOTE is always higher. NN algorithm work perfectly well on this dataset by combining and recalculate weight between feature for reducing the cost after each training. For improving performance of KNN, SVM, RF we must combine features to create a better data for classification.