Term Deposit Classifier - Machine Learning Capstone Project

Wenke Huang

September 8, 2017

1. Introduction

Financial problem is one the serious problems that may drag people from heaven back to earth. For example, some people may quit college because of the tuition or supporting the family. We may realize that a term deposit which is like a self-funded insurance and a back-up plan may help avoid or reduce the influence of them.

We would like to build classifiers based on supervised learning algorithms to make classification about if a client has subscribed a term deposit. We also would like to analyse the relationship of different features, and if the observations can be clustered into several groups.

The dataset is obtained from UCI Machine Learning Repository and contains 45211 observations and 17 variables (including response and predictors).

2. Data Exploration

2.1 Response

The response variable `y` has two values "yes" and "no" here, which stands for whether the client has subscribed a term deposit or not, respectively. We would like to use the frequency table below to show the general distribution of our response variable.

y	yes	no	total
Freqency	5289	39922	45211
Percentage	11.7%	88.3%	100%

Obviously, our response variable `y` here does not follow a balanced distribution. Because around 88.3% of the total observations have value "no" while only the rest 11.7% have value "yes". Luckily, there is no missing values here.

2.2 Removed Variables

There are two variables `day` and `month` in our dataset which stand for last contact day of month and last contact month of year, respectively. I think that these two variables should be considered as categorical variables even though variable `day` were recorded by using numerical values. Because the values of these two variables do not actually have the corresponding quantitative meanings here. Instead, these two are kind of the time index of the last contact time.

Hence, I would like not to take these two variables into consideration for the rest of this project. We may also consider removing additional variables in the following parts.

2.3 Continuous Variables

There are 6 continous variables in our dataset. They are "age", "balance", "duration", "campaign", "pdays" and "previous". The description and the scatter matrix plot with diagonal being density plot of them are shown below.

- age: how old the client is?
- balance: the current balance in the bank
- duration: last contact duration, in seconds (numeric). **Important note**: this attribute highly affects the output target (e.g., if duration=0 then y="no"). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.
- campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
- pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric)
- previous: number of contacts performed before this campaign and for this client (numeric)

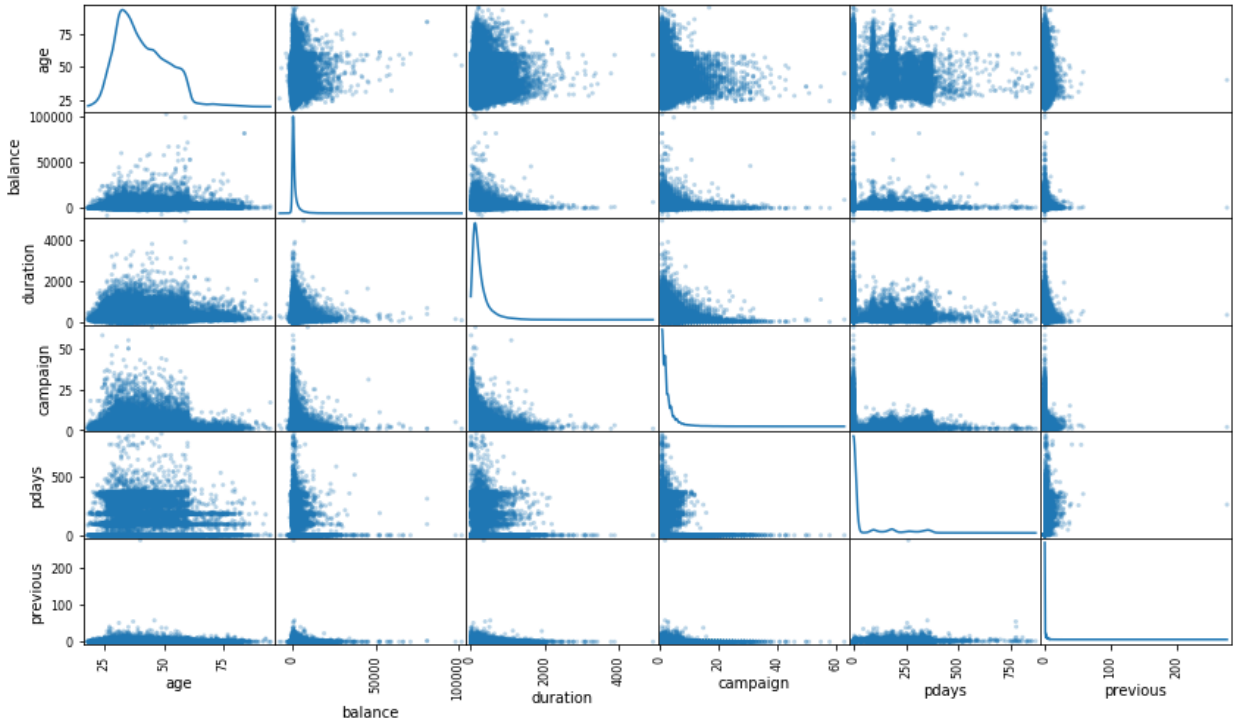

Figure 1: Scatter Matrix Plot of Continuous Variables

From the description and plot above, I have several observations here.

1. Variable `duration` is highly correlated with our response variable `y`. If `duration` is 0, then `y` has to be "no".
2. It is very hard to visually detect any obvious linear correlationship between different groups of two variables. The distributions of all these six variables are positively skewed.
3. There is one observation with extremely high `previous` value. We would like to remove that point to lower the influence that may be brought from that observation. The new scatter plot is shown below, and it looks much better.

Since we remove only 1 observation, the distribution of our response variable `y` will not change dramatically.

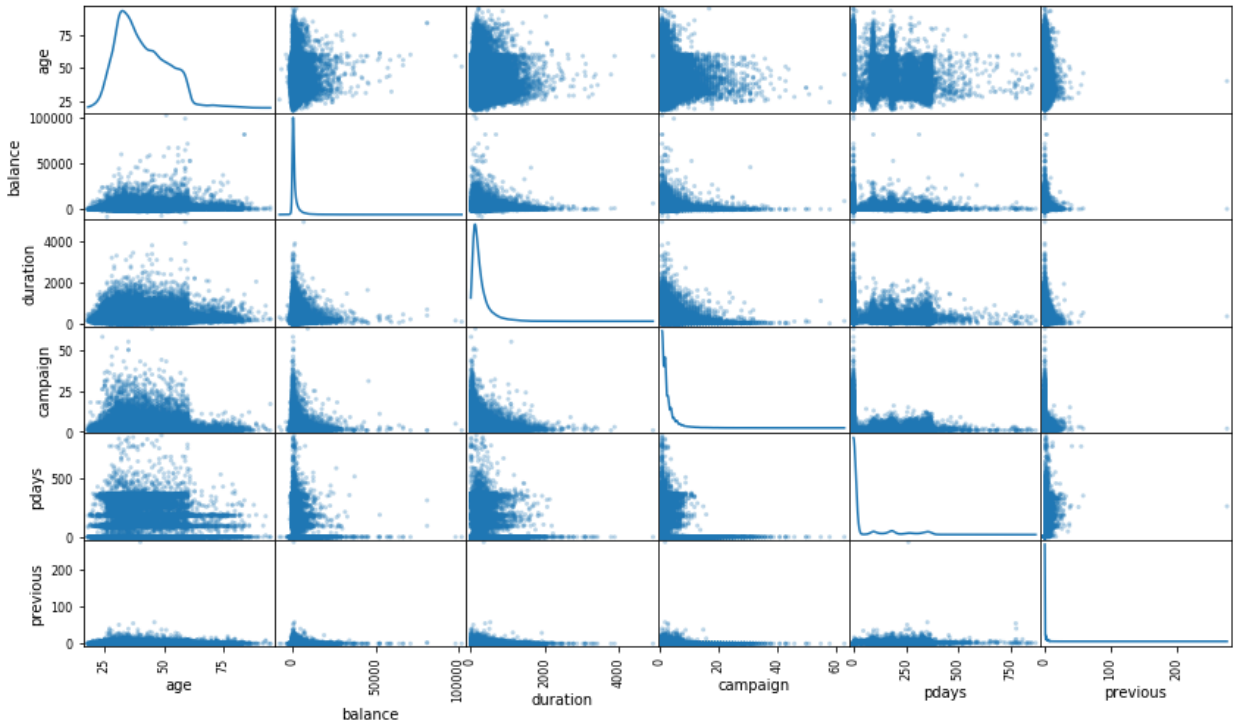

Figure 2: Scatter Matrix Plot of Continuous Variables (without outlier)

It will still be an unbalanced distribution.

2.4 Categorical Variables

The rest 8 variables in our dataset are all categorical variables. They are "job", "marital", "education", "default", "housing", "loan", "contact" and "poutcome". The description and barplots of them are shown below.

- job: type of job ("admin.", "blue-collar", "entrepreneur", "housemaid", "management", "retired", "selfemployed", "services", "student", "technician", "unemployed", "unknown")
- marital: marital status ("divorced", "married", "single", "unknown"; note: "divorced" means divorced or widowed)
- education: education level ("secondary", "tertiary", "primary", "unknown")
- default: has credit in default? ("no", "yes")
- housing: has housing loan? ("no", "yes")
- loan: has personal loan? ("no", "yes")
- contact: contact communication type ("cellular", "telephone", "unknown")
- poutcome: outcome of the previous marketing campaign ("failure", "other", "success", "unknown")

From the description and plots above, I have a couple of observations here.

1. Variable `poutcome` contains more than 35000 missing values here. Even though as a categorical variable, missing values can be considered as another category. I strongly do not think that the frequency of missing values is as large as this is a good idea. Hence, variable `poutcome` will not be considered in the following parts.
2. There are still other variables that have missing values here, like `contact`, `education` and `job`. We will try to treat those values as another category of the corresponding variables.

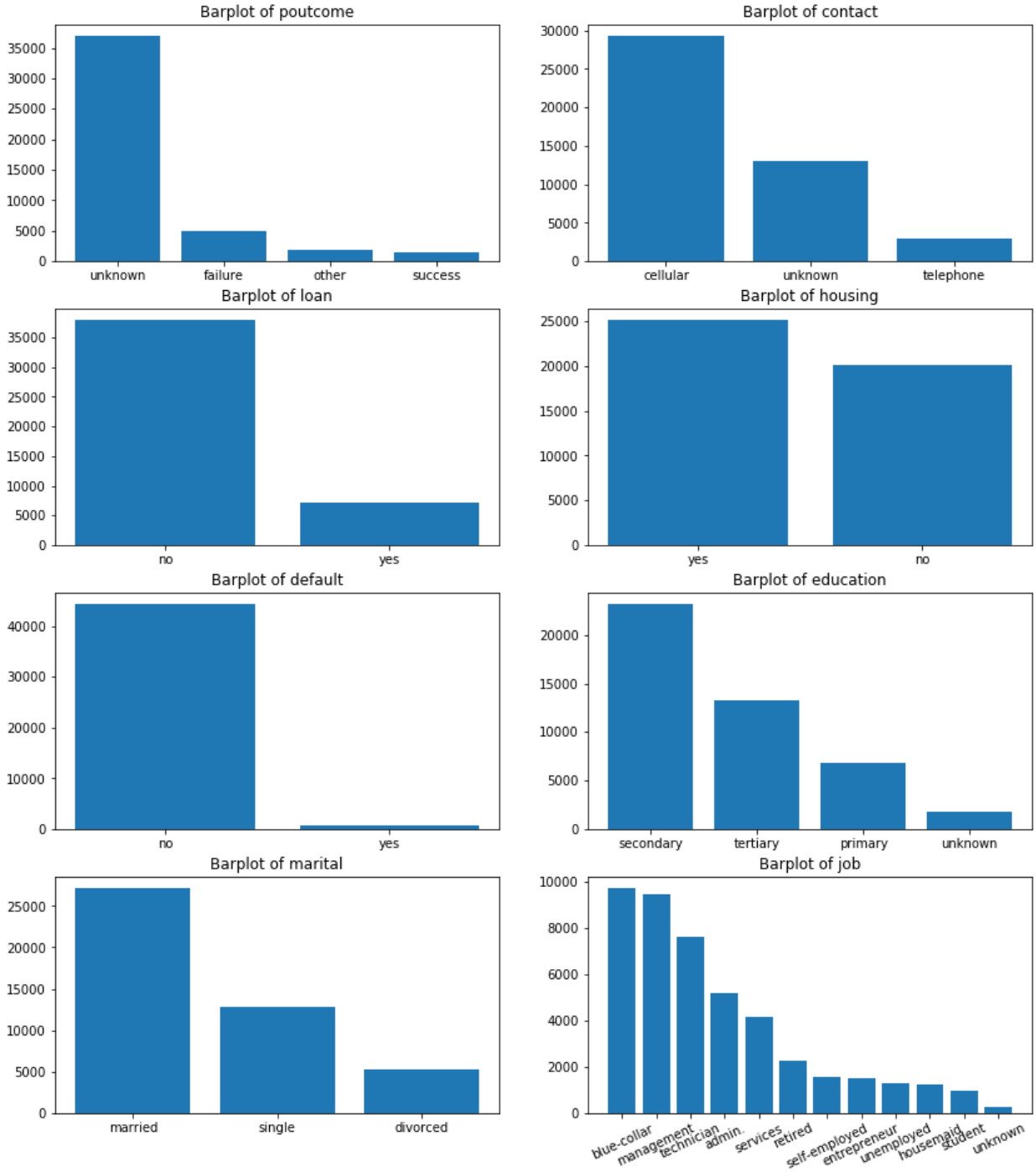

Figure 3: Barplots of Categorical Variables

2.5 Summary

In this part, we removed three variables `day`, `month` and `poutcome` and one observation with extremely high `previous` value. The dataset now contains 6 continuous variables, 7 categorical variables and 45120 observations in total.

3. Preprocessing

3.1 Outliers and Invalid Values

As we have mentioned above, there is one observation with extremely high `previous` value and variable `poutcome` contains more than 35000 missing values which were recorded as "unknown".

For the extreme `previous` value, I would like to remove this point from our dataset. For the variable `poutcome`, I would like to remove the whole variable entirely. We do notice that there are some other categorical variables with "unknown" values. Since the proportions of them are not extremely large, we would like to treat "unknown" value as a new category of the corresponding variables.

3.2 One-hot Encoding and Min-max Normalization

One-hot encoding here is used to transform the 7 categorical variables and response variable `y` into numerical ones. For one particular variable `var`, the number of new variables `var_new` after being one-hot encoded is equal to the level of this variable `var` minus one.

Min-max Normalization is used to transform our 6 continuous variables. The formula is $new_var = \frac{old_val - min(old_val)}{max(old_val) - min(old_val)}$. The ranges of our new features will all be from 0 to 1.

Particularly, for the response variable `y`, value "no" and "yes" will be converted to 0 and 1, respectively.

```
old_y|new_y
  no |  0
  no |  0
  no |  0
 ...   ...
 yes |  1
  no |  0
  no |  0
```

3.3 Metric

`Accuracy`, `precision` and `recall` are the metrics that we are about to use here. All these three metrics range from 0 to 1. The higher the values is, the better our model is. The formulas of them are shown below:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$precision = \frac{TP}{TP + FN}$$

$$recall = \frac{TP}{TP + FP}$$

where `TP`, `TN`, `FP` and `FN` stand for True positive, True negative, False positive and False negative, respectively. We can read that `accuracy` mainly focuses on the overall correct classification for both positive and negative label while `precision` and `recall` concentrate on correct classification for positive label (in this project it means `y` equal to "yes") only.

3.4 Feature Selection

We have 13 features in our project, but we may not want to use all of them. Because too many predictors may lead to over-fitting which indicates that our models only work well on training set instead of testing set. We absolutely do not want that happen. Therefore, before we jump into the next part, we would like to select the features that we are about to use.

The method that I would like to choose is `Decision Tree` algorithm with the "Gini" splitting criterion. After we build the model with all features as predictors, I would like to calculate their variable importance, and visually present the results by using a horizontal bar plot.

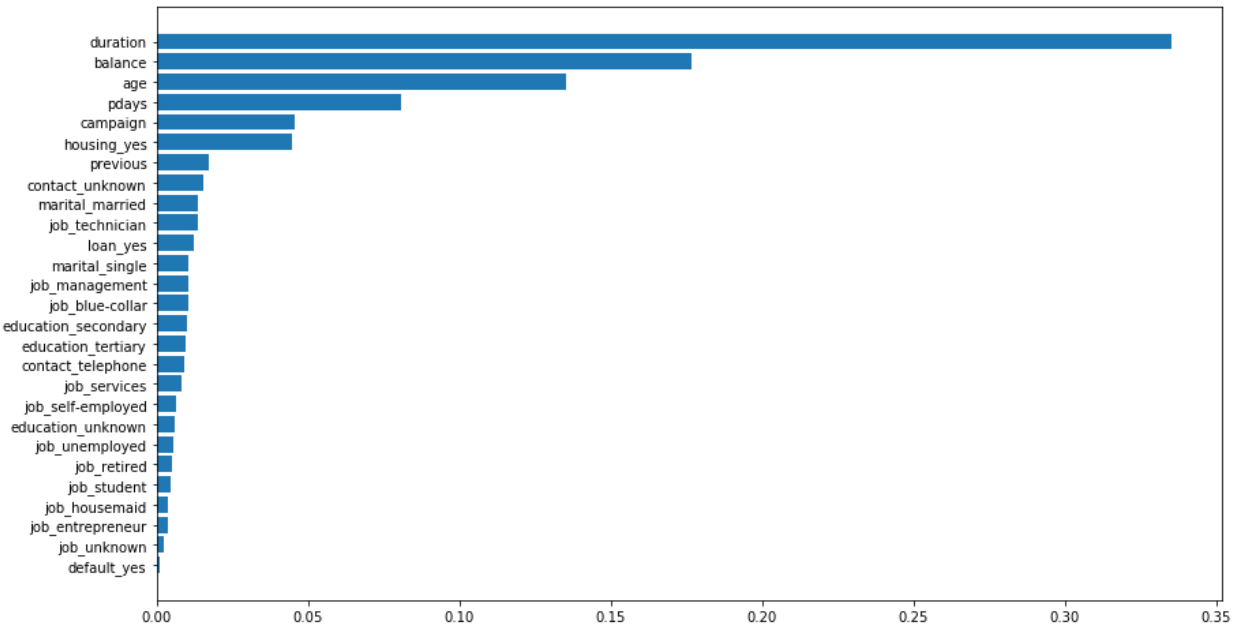

Figure 4: Variable Importance

We can find out that variable `duration`, `balance`, `age`, `pdays`, `housing` (binary variable, housing_yes in the plot) and `campaign` obviously greater than other variables. There also seems to be an "elbow" which divide these six variables from the rest. Hence, I would like to use these six variables as our final predictors to build our classifiers.

3.5 Training and Testing Splitting

We would like to split the whole dataset into training and testing subsets randomly. Here, we also will use stratified split here since the distribution of our response variable `y` does not follow a balanced distribution. The training set contains 36168 (80%) observations and the testing set contains the rest 9042 (20%) ones.

y	yes	no	total
Training Frequency	31937	4231	36168
Training Percentage	88.3%	11.7%	100%

y	yes	no	total
Testing Frequency	7984	1058	9042
Test Percentage	88.3%	11.7%	100%

The distribution of response variable `y` of both training and testing sets are shown above. The percentages of "yes" and "no" of both training and testing are basically equal to the corresponding percentages of the whole dataset.

4. Model Fitting

4.1 Benchmark Model

Ususally, older people tend to have a higher willing to subscribed term deposit since they are less open to the risk because of family, physical issues and/or some other reasons. Hence, our benchmark model would be set based on variable age only. If age is greater than 65, we predict those people have subscribed term deposit already. If age is less than or equal to 65, we don't think they have subscribed it yet.

4.2 Logistic Regression

First, let us try to use `Logistic Regression` algorithm. `Logistic regression` is a regression model where our response variable is categorical. In order to restrict range of the output within 0 and 1, we will like to add a logit link function to transform from y to $\frac{1}{1+e^{-y}}$. The performance of this algorithm may not be very good, but we are able to tell the quantitative relationship between predictors and response variable.

4.3 K Nearest Neighbors

Then, I would like to try to use `K Nearest Neighbors` algorithm. For this algorithm, we calculate the distance between one point and its closest `k` different points, and then use majority vote to classify our response variable `y`. Minkowski distance with p equal to 2 which is also called Euclidean distance will be used here. We would like to use 5-fold stratified cross-validation to tune the parameter `k`, and we find out the `k` with the best performance is `7`.

4.4 Random Forest

`Random Forest` is a very powerful machine learning algorithm. It ensembles single tree models together and provides stronger classification. But in most case, it tends to over-fit the dataset due to its flexibility. Like the previous part, we will us 5-fold stratified cross-validation to tune the parameters `ntree` and `mtry` which stands for number of trees and attempt variables. The parameters with the best performance here is `ntree =` and `mtry =`

4.5 Multilayer Perceptron Neural Network

`MLP Neural Network` consists three layers of nodes (the input layer, at least one hidden layer, and the output layer). For the hidden layers, we will choose to use the ReLU activation function. In order not to over-fit our model, the probability of each hidden layer node being dropped is set to be 0.2. Here, we will have two hidden layers and the learning rate is 0.001. Training epoch is 20.

We know that if the learning rate is small, we may just stop at the local minima. Hence, I would also like to fit another model with a larger learning rate (0.05). Everything else will remain the same.

5. Summary

The performances of the previous five models are shown below. For algorithm `K Nearest Neighbors` and `Random Forest`, we only list the ones with the best performances. For algorithm `Multilayer Perceptron Neural Network`, the result of both learning rates will be listed.

Algorithm	train Accuracy	test Accuracy	train Recall	test Recall	train Precision	test Precision
Benchmark	88.13%	88.43%	3.88%	5.10%	41.93%	56.25%
Logistic Reg.	88.97%	88.89%	17.16%	15.97%	59.90%	59.30%
KNN (k=11)	90.62%	89.23%	38.48%	32.70%	67.38%	56.91%
RF (ntree=300, mtry=2)	99.99%	89.66%	99.98%	36.96%	100.00%	59.33%
MLP N. Net (lr=.001)	89.11%	88.97%	29.90%	30.06%	57.37%	60.23%
MLP N. Net (lr=.05)	89.10%	89.37%	24.25%	24.76%	58.06%	61.36%

Here, we mainly focus on the testing accuracy, recall and precision.

Even though the overall accuracy of our benchmark model is basically equal to the accuracies of the rest five models, the recall score is much lower than others and the precision is slighly worse than others'. We may think that the preformance of our benchmark model is OK, and the rest five are all more satisfactory than it. Obviously, the training accuracy, recall and precision of `Random Forest` algorithm are the best and extremely close to 100%. But the corresponding testing metrics are basically not different from the other testing metrics. Here, we can think that this algorithm over-fit our dataset. For the two `MLP Neural Network` models, the results are a little unexpected. Because in my opinion, Neural Network is a good algorithm and sometimes may over-fit our dataset like `Random Forest` which we just talked about. But the performances of them here are just somewhere between `Logistic Regression` and `KNN`.

The performance of `Logistic Regression` is slightly worse than `MLP Neural Network` while that of `KNN` is a little better than `MLP Neural Network`. For these two algorithms and benchmark model, we can quantitatively explain the relationship between different features, which is their advantage. For the rest two algorithms, it is impossible to interpret the relationship but we usually tend to get better performances. The best model for different datasets should depend on different situations.

We can learn from this that there is no universal best algorithm that works well on every dataset. Like what we have discussed in the previous paragraph, each algorithm has their own advantages and disadvantages. For this dataset, even though the training metric values of **Random Forest** are almost perfect, their testing metric values are also much better than others. Hence, we may still choose this model as our final and best model.

6. Future Thoughts

6.1 Running Time

We may think consider adding running time as one of our metrics, too. Because it usually takes much longer time for complicated models like ensemble methods to get the results. If the performance of a simple model and a complicated one are basically the same, it may not be worth using the complicated one.

6.2 Tuning Parameters

We know that as model goes more complicated, the training error is keeping going down but testing error will go down first but then go back up. We may also think manually using "imperfect" tuning parameters of models like `Random Forest`. The cross-validation will give us the one with the best performance. But that one may already over-fit. If we choose "imperfect" parameters manually, we may be able to find another model with slightly worse training error but better testing error.

Reference

- Random Forest, Wikipedia, https://en.wikipedia.org/wiki/Random_forest
- Machine Learning Nanodegree, Udacity, https://www.udacity.com/
- Precision and recall, Wikipedia, https://en.wikipedia.org/wiki/Precision_and_recall