EE559 Final Project: Develop a Pattern Recognition System for Adult Data Set by Using Machine Learning Algorithms

Data Set: Adult

Ming-Jui Lee (ID: 4673484700), mingjuil@usc.edu

Date: 4/30/2019

1. Abstract

This project aims to use pattern recognition for data classification by applying machine learning algorithms. In this project, four algorithms including perceptron, SVM, k-NN, and Naïve Bayes are used to deal with Adult data set. The purpose of this project is to predict a person's income is greater than 50K or not based on given data with both numerical and categorical features. This project is completed by following the procedure of data preprocessing, feature selection, model training, and performance evaluation. According to the result, is the best classifier is SVM with classification accuracy on testing data = 0.843928505620048, F1-score = 0.63, and AUC = 0.894.

2. Introduction

2.1. Problem Statement and Goals

In this course project, "Adult" data set is used. The goal of this project is to apply pattern recognition to predict one's income is greater than $50K or not based on given data with numerical as well as categorical features.

2.2. Literature Review

In the Adult data set, there exist missing values and the features space is consist of numerical and categorical features. People dealing with this kind of data set usually use method such as One-Hot Encoding or Label Encoder, and there are several ways to handle missing data.

3. Approach and Implementation

3.1. Preprocessing

Data preprocessing is a technique that transfers raw data into a more understandable format [1]. Before I start data preprocessing, I first replace

feature columns labels f1, f2, …, f15 with ‘age’, ‘workclass’, …, and ‘wage_class’ to make data more readable. I also find out that class label (column: ‘wage_class’) for training and testing data have different expression (<=50K and <=50K.), so I assign them with same expression. Then, I do data preprocessing with the following steps.

Step 1: Dealing with missing data

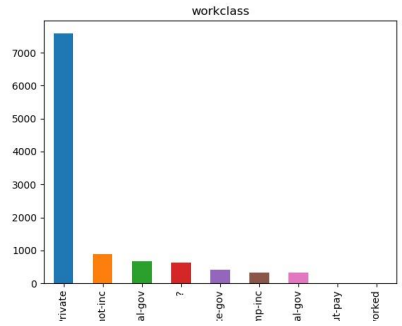
To deal with missing data, I first replace all ‘ ?’ with NaN and then drop all missing values. However, comparing to the result with keeping all missing values, the accuracy decreases slightly. After achieving data set’s basic information, I find out all missing values present in three categorical features: ‘workclass’, ‘occupation’, and ‘native_country’. Next, I try to replace missing values in these three features with its mode. Nevertheless, the accuracy on training data does not change too much. I think replacing missing values in categorical features with its mean or mode is not applicable. After consideration, I decide to keep all missing values and view ‘ ?’ as one particular feature after one-hot encoding. In addition, I assign categorical value ‘ Armed-Forces’, ‘ Hungary’, and ‘ Scotland’ to ‘ ?’ because testing data does not include these three categorical values.

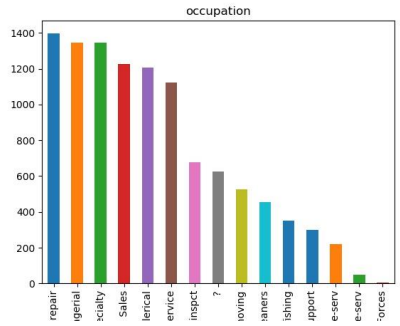
[Table 1] Removing or keeping missing data

	Learning Algorithm	Accuracy on training data
Remove missing data	Perceptron	0.7869686689283576
Keep missing data	Perceptron	0.7938818759789919

[Table 2] Replacing ‘ ?’ with mode

	Learning Algorithm	Accuracy on training data
Replace ? with mode	Perceptron	0.7938818759789919





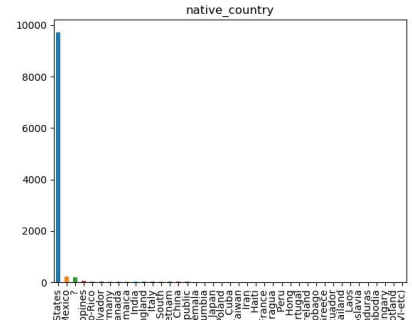


Figure (1). Mode for feature ‘workclass’, ‘occupation’, and ‘native_country’

Step 2: Re-casting different variable types

The ‘Adult’ data set is consist of numerical and categorical features. First, I use LabelEncoder function from sklearn.preprocessing to encode value 0 to category ‘ <=50K’ and value 1 to category ‘ >50K’. Then, I drop the column ‘wage_class’ because this column now represents the class label in this 2-class classification problem. After dropping the class label, there now remain six numerical features and eight categorical features. I can also use LabelEncoder to deal with other categorical features. However, this method might cause faulty classification result. For example, I encode label 1 to ‘cat’, label 2 to ‘dog’, and label 3 to ‘fish’. However, after doing data normalization, the mean of ‘cat’ and ‘fish’ would become ‘dog’. Moreover, the higher value does not necessarily mean a better category. To avoid this problem, I apply One-Hot Encoding that converts categorical variables into binary values 0 and 1, which could be provided to the learning algorithm to do a better job in prediction. To apply One-Hot Encoding, I use pandas get dummies function to create dummy variables. For a categorical feature with n categories, this feature column will be expanded to n-columns (including ‘ ?’).

Step 3: Data normalization

Intuitively, we can only apply data normalization on numerical features. The reason for doing data normalization is that features usually have a different order of magnitude. After converting these features numerical values into a given range, we can compare features and interpret results more easily. There are two primary methods to do data normalization.

Normalization: Normalization converts numerical features with different range into the unit range [0, 1]. This could be done by using MinMaxScaler from sklearn libiary. After rescaling, the meaning from the raw data will be changed (For example one’s age could not be 0.35). However, we can still

treat each feature equally. I primarily use this method in the project because all numerical features are continuous.

Standardization: Standardization makes features zero mean and unit standard deviation. We can use "StandardScaler" from sklearn library to complete data standardization.

[Table 3] Compare classification accuracy with and without normalization

	Learning Algorithm	Accuracy on training data
With normalization	Perceptron	0.826591725790104
Without normalization	Perceptron	0.7938818759789919

We only train our classifier based on the training data and cannot use testing data set in advanced. Consider testing data set that we have never seen before. Then we definitely do not know its mean or other normalizing factors. If we use testing data in advance, it will become data snooping. We should never do data snooping before training classifier.

Step 4: Data values combination

In this step, I try different ways to combine features values in numerical features as well as in categorical features and find out which way to manipulate data can provide me with better results. In this part, I handle each feature separately and use perceptron learning to get accuracy on training data.

[Table 4] Handling each class separately

Feature	combination	Accuracy on training data
	Original values	0.826591725790104
Age (numerical)	Assign all values to nine intervals	0.8185755090758316
Workclass (categorical)	Combine some workclass categories into one category	**0.8346079425043766**
Education (categorical)	Combine some education level into one category	0.8233668110199945
Marital_status (categorical)	combine Married-civ-spouse and Married-AF-spouse to Married	0.7992260204551737
Occupation (categorical)	Combine some occupations based on Major Occupational Groups (MOGs)[3]	**0.8310144660462545**
Native_country	combine countries with same	0.800147424675205

(categorical)	income level based on List of countries by GNI (nominal) per capita[4]	

[Table 5] Handling more than one feature in the same time

Workclass (categorical) and Occupation (categorical)	Combine some workclass categories into one category and combine some occupations based on Major Occupational Groups (MOGs)[3]	0.8017138118492582

According to the results of accuracy on training data, I decide to keep data values as intact as possible because doing such combination does not contribute significantly to accuracy improvement.

3.2. Compensation for unbalanced data (if applicable)

For this project, I do not handle the unbalanced data.

3.3. Feature extraction and dimensionality adjustment

For numerical feature selection, I decide to use one simple method for 2-Class problem by following below steps.

Steps:

1. Standardize all Training data
2. Train a linear classifier
3. Find weight for each feature. The feature i with largest |wi| contribute the most to the classifier decision.

Following these steps, I train a perceptron linear classifier by using numerical features. The corresponding weights to each numerical feature are shown in the below figure.

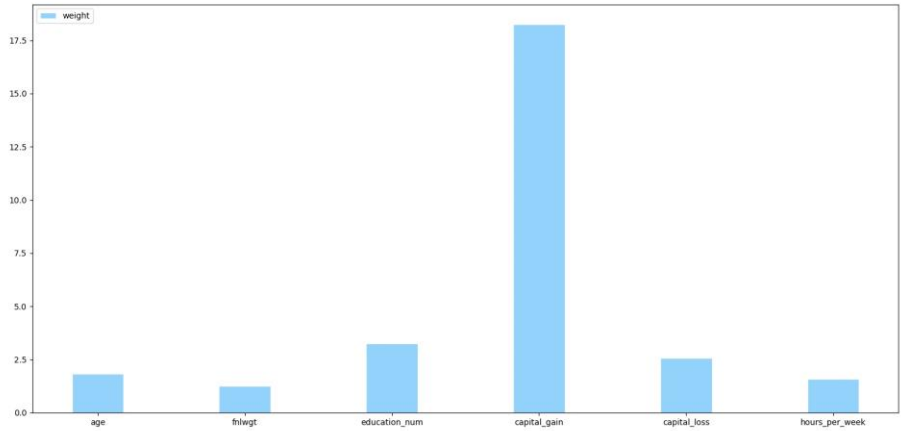


Figure (2). Weights corresponding to each numerical feature

According to the result, I first decide to drop feature 'fnlwgt' first because it has the lowest weight and does not provide any useful or interpretable information to me. As we can see, after dropping feature 'fnlwgt', the accuracy on training data remain almost the same.

[Table 6] Keep or drop feature 'fnlwgt'

	Learning Algorithm	Accuracy on training data
Original data	Perceptron	0.826591725790104
Drop 'fnlwgt'	Perceptron	0.8225375472219663

Secondly, I decide to drop feature 'education' because it seems to provide the same information as the numerical feature 'education_num' does.

[Table 7] Drop vs. Keep feature

	Learning Algorithm	Accuracy on training data
Original data	Perceptron	0.826591725790104
Drop 'fnlwgt' and 'education'	Perceptron	0.8052151478853773

After doing feature selection and dimensionality adjustment, the accuracy in training data decreases slightly. That is to say, when using 87 features rather than 104 features to train the classifier, we can get similar result and enhance the efficiency at the same time.

3.4. Dataset Usage

In the project, I use 10853 training data points and 5427 testing data points.

I first add feature columns name for both training and testing dataset and get basic data information from both datasets. Then, I start to do data preprocessing and feature selection.

For cross-validation, I use Stratified K-Folds cross-validator. For this cross-validation technique, the original dataset will be split into k subsets with almost the same ratio of 2-class (in Adult dataset class 0:class 1 approximately equal to 3:1) in original dataset. The number of fold I used in this project is five.

For SVM parameter selection, I run cross-validation for one nested loop and choose gamma and C pair based on best accuracy on training data. I use cross-validation to check 15 points in the range from 0.01 to 1. For KNN model, I run cross-validation to choose the value of k with highest accuracy on training data.

Testing dataset is used for five times after all classifiers are trained.

3.5. Training and Classification

In this project, I use training data to train four classifiers: Perceptron, SVM, KNN, and Naïve Bayes.

3.5.1 Perceptron

- In machine learning, the perceptron is an algorithm for supervised learning of binary classifiers [5]. Perceptron is an algorithm for learning a binary linear classifier called threshold function f(x): this function takes an input sample **x** (vector with real value) and map it to a binary output value. That is to say, the function decides the input **x** belong to class 1 or class 0.

  f(x) = 1 if **w** ∘ **x** +w0 **w** is a real-valued weight vector for input **x**

  = 0 otherwise w0 is the bias, **w** ∘ **x** denotes the inner product

  Then, the perceptron learning algorithm updates the weight vector based on the following equations:

  (ii) Perform single-sample update

$$\begin{cases} \underline{w}(i+1) = \underline{w}(i) + \eta(i) z_n \underline{x}_n & \text{if } \underline{w}(i)^T z_n \underline{x}_n \le 0 \\ \underline{w}(i+1) = \underline{w}(i) & \text{if } \underline{w}(i)^T z_n \underline{x}_n > 0 \end{cases}$$

  z = -1 when the data point **X**i is misclassified by the current weight vector; z = 1 means that the data point is correctly classified by the

weight vector. We iterate the perceptron algorithm until there is no training data point is misclassified by the current weight vector.

In python, the perceptron learning algorithm is implemented from sklearn.linear_model import Perceptron. I use default fixed increment eta = 1. I manually set parameters tol=1e-3, random_state=0.

3.5.2 Support Vector Machine

- Support Vector Machine is a supervised learning model that assigns given data points to either class 1 or class 0. For the 2-Class problem, we would like to find the hyperplane that can make two nearest points from each class have largest distance. We call this hyperplane "maximum-margin hyperplane." The data point that lies on the margin is known as "support vector." We can also apply kernel function to train SVM as a non-linear classifier. From the library sklearn.svm, we can use SVM with "from sklearn.svm import SVC."

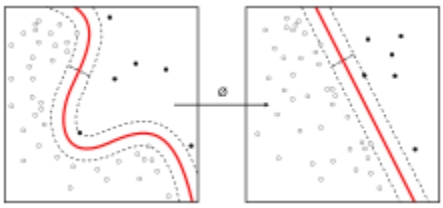


Figure (3). SVM[6]

For parameter setting, I set kernel = 'rbf', which denotes radial basis function. The Gaussian (RBF) Kernel transforms the original feature space to higher dimensional new feature space, the decision boundary we get when using RBF kernel will become non-linear. Then, I use cross-validation to select parameter **gamma = 0.0439397** and **C = 10**. The parameter C tells the SVM classification how much the penalty we want for misclassifying each training data point. Adjusting C provides a method to control overfitting. When C is small, the classifier can tolerate misclassified data points (decision boundary with larger margin). On the contrary, for large C, the classifier is heavily penalized for misclassified data, so we will get a decision boundary with a smaller margin. Gamma is the parameter for the Gaussian (RBF) kernel and can be viewed as the degree of spread of the kernel and decision region. When gamma is small, the curve of the decision boundary is very low and thus the decision region is very broad. As gamma getting larger, the curve of the decision boundary becomes higher, this generates island-like decision boundaries around data points. The behavior of the model is very sensitive to the gamma parameter. If

gamma becomes too large, the radius of the decision region only includes the support vector itself. Therefore, the decision boundary will almost depend on an individual data point and the data will easily be overfitted.

3.5.3 K-nearest neighbors

- In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression [7]. The output of k-NN classification is a binary value 1 or 0. For k-NN classification, the input vector **x** is center at the region with k nearest training data samples. This vector **x** is then classified by a plurality vote of its k nearest neighbors, and assigned to the class that is a majority of its k nearest neighbors. When k is equal to 1, we will have to face the over-fitting problem. If k is equal to N (number of training data point), the input will always be assigned to the class most common in this training dataset. That is to say, we do not do any training on the classifier. For 2-Class problem, we usually choose an odd value k as a tiebreak.

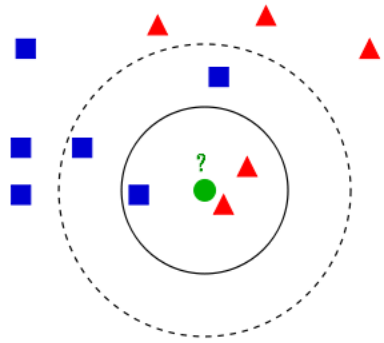


Figure (4). k-NN algorithm [8]

For k-NN model, I use five folds cross-validation by using "from sklearn.model_selection import StratifiedKFold", and use "sklearn.model_selection.cross_val_score" to evaluate a score by cross-validation. Then, I choose a value of k with highest cross validation score for the k-NN model. According to cross-validation, I choose **k = 9**. For k-NN model, I use "from sklearn.neighbors import KNeighborsClassifier" and set n_neighbor = 9.

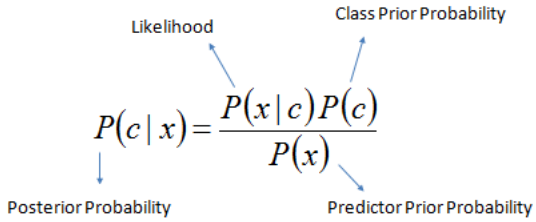
[Tabel 7] Selecting k by cross-validation

K	1	2	3	4	5
Cross-validation score	0.79305	0.80428	0.81378	0.81452	8.82281

K	6	7	8	**9**	10
Cross-validation score	0.82078	0.82447	0.82336	**0.82815**	8.82714

3.5.4 Naive Bayes classifier

- Naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features [9]. To put it more clearly, we assume that one feature has nothing to do with other features. Moreover, we treat all features with equal importance. We can apply Bayes Theorem to compute posterior probability from likelihood, class prior probability, and predictor prior probability, as figure (5) shown below. To implement Naïve Bayes classifier, I use "from sklearn.naive_bayes import GaussianNB".



$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Figure (5). Bayes Theorem [10]

Normalization is not necessary for Naïve Bayes model because we only care about data's distribution. Therefore, I use data set with basic data preprocessing and without normalization for Naïve Bayes classifier.

4. Comparison, Results, and Interpretation

For the "Adult" dataset, the baseline is a classifier that always decides the majority class, which is class 0 (<=50K). The baseline for the Adult dataset is a classifier that always decides the majority class (class 0: <=50K). We can calculate the **baseline accuracy = 0.75923** from the training data set. Observing the classification accuracy on testing data, all of these four classifiers can get some gain on classification accuracy. After data preprocessing and feature selection, perceptron, SVM, and k-NN model can get even better performance on classification. The performance of the Naïve Bayes model remains unchanged because it treats all features independently and makes a classification based on the distribution of data points.

[Tabel 8] Accuracy on dataset before preprocessing and feature selection

Model	Accuracy on training data	Accuracy on testing data
Perceptron	0.7938818759789919	0.7960199004975125
SVM(gamma=0.0439397, C= 10)	0.9510734359163365	0.8369264787175235
k-NN(n_neighbors=9)	0.8100985902515434	0.785148332412014
Naïve Bayes	0.7980281949691329	0.796941219826792

[Tabel 9] Accuracy on dataset after preprocessing and feature selection

Model	Accuracy on training data	Accuracy on testing data
Perceptron	0.8052151478853773	0.8116823290952644
SVM(gamma=0.0439397, C=10)	0.8525753247949875	**0.843928505620048**
k-NN(n_neighbors=9)	0.8566295033631254	0.8264234383637369
Naïve Bayes	0.7922233483829356	0.7844112769485904

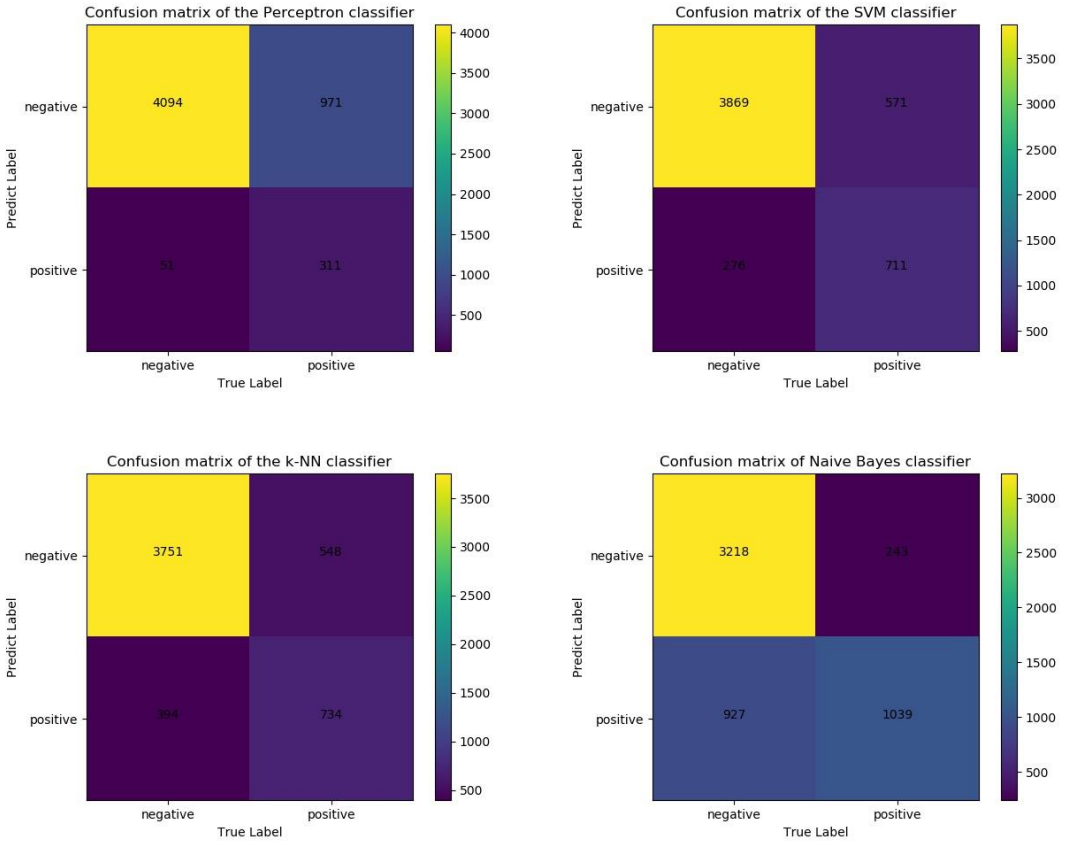


Figure (5). Plots of confusion matrix

[Table 9] Precision, Recall, and F1-score

Model	Precision	Recall	**F1-score**	AUC
Perceptron	0.75	0.37	**0.49**	0.860
SVM(gamma=0.0439397, C=10)	0.72	0.55	**0.63**	**0.894**
k-NN (k=9)	0.65	0.57	**0.61**	0.858
Naïve Bayes	0.53	0.81	**0.64**	0.842

When doing 2-class classification, it would be too arbitrary to evaluate a classifier's performance only based on classification accuracy. For example, there is an unbalanced dataset with 95 samples belong to class 0, and 5 samples belong to class 1. If I always decide a sample with class 0, I can get a model with 95% classification accuracy. However, if I am more interested in minority class 1, the 95% classification accuracy is not useful to me. To avoid this condition and better evaluate the model performance, more measure metrics need to be considered.

By using "from sklearn.metrics import confusion_matrix", we can obtain confusion matrix. To plot confusion matrix, I refer the code from the question from stackoverflow. With the values of True-Positive, False-Positive, True-Negative, and False-Negative, we can calculate precision, recall, and F1-score. For purposes of the F1 score, we define the minority class (class 1: >50K) as the positive class in this project. To get precision, recall, and F1-score, I use "from sklearn.metrics import classification_report". We know that F1-score is harmonic mean of precision and recall, and can be used to evaluate the performance of the model. In addition, I apply "from sklearn.metrics import roc_auc_score" to calculate AUC (**A**rea **U**nder the Receiver Operating Characteristic **C**urve), which is also is a measure of how well the model is. According to the classification accuracy, F1-score, and AUC, we conclude that SVM model has the best performance of classification among the four models I have used.

5. Summary and conclusions

To sum up, we can get better classification performance after doing feature selection as well as proper data preprocessing. For choosing values of parameters, Cross-validation is applied to select parameter gamma and C for SVM and k value for k-NN. According to the result of accuracy metrics, SVM classifier presents the best performance for the prediction of one's income is greater than $50K or not.

References

[1] “What Steps should one take while doing Data Preprocessing?” 20 April 2019. [Online]. Available: https://hackernoon.com/what-steps-should-one-take-while-doing-data-preprocessing-502c993e1caa

[2] “What is One Hot Encoding? Why And When do you have to use it?” 25 April 2019. [Online]. Available: https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f

[3] “National Compensation Survey - Wages” 25 April 2019 [Online]. Available: https://www.bls.gov/ocsm/commain.htm

[4] “List of countries by GNI (nominal) per capita” 25 April 2019 [Online]. Available: https://en.wikipedia.org/wiki/List_of_countries_by_GNI_(nominal)_per_capita

[5] “Perceptron” 25 April 2019 [Online]. Available: https://en.wikipedia.org/wiki/Perceptron

[6] “Support-vector machine” 25 April 2019 [Online]. Available: https://en.wikipedia.org/wiki/Support-vector_machine

[7] “k-nearest neighbors algorithm” 25 April 2019 [Online]. Available: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[8] “Understand k-nearest neighbors algorithm” 25 April 2019 [Online]. Available: https://zhuanlan.zhihu.com/p/25994179

[9] “Naive Bayes classifier” 25 April 2019 [Online]. Available: https://en.wikipedia.org/wiki/Naive_Bayes_classifier

[10] “Naive Bayes classifier” 25 April 2019 [Online]. Available: https://mropengate.blogspot.com/2015/06/ai-ch14-3-naive-bayes-classifier.html

[10] “Naive Bayes classifier” 25 April 2019 [Online]. Available: https://mropengate.blogspot.com/2015/06/ai-ch14-3-naive-bayes-classifier.html

[11] “sklearn plot confusion matrix with labels” 26 April 2019 [Online]. Available: https://stackoverflow.com/questions/19233771/sklearn-plot-confusion-matrix-with-labels/31720054