Iqra National University Peshawar Pakistan

Department of Computer Science

Assignment No 1

Fall Semester 2020

Subject: Machine Learning

Submitted by: Noor Rahman

Registration ID: 14232

Program: MSCS

Submit to: Dr: Fazal-e-Malik

Iqra National University Peshawar Pakistan

Department of Computer Science

Q 1) There are several machine learning algorithms for classification. The following are some of the well-known algorithms.

1. Logistic regression

Logistic regression is a statistical analysis method used to predict a data value based on prior observations of a data set. Logistic regression has become an important tool in the discipline of machine learning. The approach allows an algorithm being used in a machine learning application to classify incoming data based on historical data. As more relevant data comes in, the algorithm should get better at predicting classifications within data sets. Logistic regression can also play a role in data preparation activities by allowing data sets to be put into specifically predefined buckets during the extract, transform, load (ETL) process in order to stage the information for analysis.

A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables. For example, a logistic regression could be used to predict whether a political candidate will win or lose an election or whether a high school student will be admitted to a particular college.

The resulting analytical model can take into consideration multiple input criteria. In the case of college acceptance, the model could consider factors such as the student’s grade point average, SAT score and number of extracurricular activities. Based on historical data about earlier outcomes involving the same input criteria, it then scores new cases on their probability of falling into a particular outcome category.

Purpose and examples of logistic regression

Logistic regression is one of the most commonly used machine learning algorithms for binary classification problems, which are problems with two class values, including predictions such as “this or that,” “yes or no” and “A or B.”

The purpose of logistic regression is to estimate the probabilities of events, including determining a relationship between features and the probabilities of particular outcomes.

One example of this is predicting if a student will pass or fail an exam when the number of hours spent studying is provided as a feature and the variables for the response has two values: pass and fail.

Organizations can use insights from logistic regression outputs to enhance their business strategies so they can achieve their business goals, including reducing expenses or losses and increasing ROI in marketing campaigns, for example.

An e-commerce company that mails expensive promotional offers to customers would like to know whether a particular customer is likely to respond to the offers or not. For example, they'll want to know whether that consumer will be a "responder" or a "non responder." In marketing, this is called *propensity to respond modeling*.

Likewise, a credit card company develops a model to decide whether to issue a credit card to a customer or not will try to predict whether the customer is going to default or not on the credit card based on such characteristics as annual income, monthly credit card payments and number of defaults. In banking parlance, this is known as *default propensity modeling*.

Uses of logistic regression

Logistic regression has become particularly popular in online advertising, enabling marketers to predict the likelihood of specific website users who will click on particular advertisements as a yes or no percentage.

Logistic regression can also be used in:

- Healthcare to identify risk factors for diseases and plan preventive measures.
- Weather forecasting apps to predict snowfall and weather conditions.
- Voting apps to determine if voters will vote for a particular candidate.
- Insurance to predict the chances that a policy holder will die before the term of the policy expires based on certain criteria, such as gender, age and physical examination.
- Banking to predict the chances that a loan applicant will default on a loan or not, based on annual income, past defaults and past debts.

Binary Logistic Regression Major Assumptions

1. The dependent variable should be dichotomous in nature (e.g., presence vs. absent).
2. There should be no outliers in the data, which can be assessed by converting the continuous predictors to standardized scores, and removing values below -3.29 or greater than 3.29.
3. There should be no high correlations (multi collinearity) among the predictors. This can be assessed by a correlation matrix among the predictors. Tabachnick and Fidel (2013) suggest that as long correlation coefficients among independent variables are less than 0.90 the assumption is met.

At the center of the logistic regression analysis is the task estimating the log odds of an event. Mathematically, logistic regression estimates a multiple linear regression function defined as:

$$\text{logit}(p) = \log\left(\frac{p(y=1)}{1-(p=1)}\right) = \beta_0 + \beta_1 \cdot x_{i2} + \beta_2 \cdot x_{i2} + \_ + \beta_p \cdot x_{in}$$

for i = 1…n.

Overfitting. When selecting the model for the logistic regression analysis, another important consideration is the model fit. Adding independent variables to a logistic regression model will always increase the amount of variance explained in the long odds (typically expressed as R^2). However, adding more and more variables to the model can result in overfitting, which reduces the generalizability of the model beyond the data on which the model is fit.

Reporting the R^2. Numerous pseudo-R^2 values have been developed for binary logistic regression. These should be interpreted with extreme caution as they have many computational issues which cause them to be artificially high or low. A better approach is to present any of the goodness of fit tests available; Hosmer-Lemeshow is a commonly used measure of goodness of fit based on the Chi-square test.

Logistic regression vs. linear regression

The main difference between logistic regression and linear regression is that logistic regression provides a constant output, while linear regression provides a continuous output.

In logistic regression, the outcome, such as a dependent variable, only has a limited number of possible values. However, in linear regression, the outcome is continuous, which means that it can have any one of an infinite number of possible values.

Logistic regression is used when the response variable is categorical, such as yes/no, true/false and pass/fail. Linear regression is used when the response variable is continuous, such as number of hours, height and weight.

For example, given data on the time a student spent studying and that student's exam scores, logistic regression and linear regression can predict different things.

With logistic regression predictions, only specific values or categories are allowed. Therefore, logistic regression can predict whether the student passed or failed. Since linear regression predictions are continuous, such as numbers in a range, it can predict the student's test score on a scale of 0 -1.

……………………………………………………………………………………….

2. Naive Bayes algorithm

Definition - What Does Naive Bayes mean?

A naive Bayes classifier is an algorithm that uses Bayes' theorem to classify objects. Naive Bayes classifiers assume strong, or naive, independence between attributes of data points. Popular uses of naive Bayes classifiers include spam filters, text analysis and medical diagnosis. These classifiers are widely used for machine learning because they are simple to implement.

Naive Bayes is also known as simple Bayes or independence Bayes.

Explains Naive Bayes

A naive Bayes classifier uses probability theory to classify data. Naive Bayes classifier algorithms make use of Bayes' theorem. The key insight of Bayes' theorem is that the probability of an event can be adjusted as new data is introduced.

What makes a naive Bayes classifier naive is its assumption that all attributes of a data point under consideration are independent of each other. A classifier sorting fruits into apples and oranges would know that apples are red, round and are a certain size, but would not assume all these things at once. Oranges are round too, after all.

A naive Bayes classifier is not a single algorithm, but a family of machine learning algorithms that make uses of statistical independence. These algorithms are relatively easy to write and run more efficiently than more complex Bayes algorithms.

The most popular application is spam filters. A spam filter looks at email messages for certain key words and puts them in a spam folder if they match.

Despite the name, the more data it gets, the more accurate a naive Bayes classifier becomes, such as from a user flagging email messages in an inbox for spam.

Naive Bayes Classifiers

Naive Bayes classifiers are a collection of classification algorithms based on Bayes’ Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

To start with, let us consider a dataset.

Consider a fictional dataset that describes the weather conditions for playing a game of golf. Given the weather conditions, each tuple classifies the conditions as fit(“Yes”) or unfit(“No”) for plaing golf.

Naive Bayes algorithm?

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:

$$\underset{\text{Posterior Probability}}{P(c \mid x)} = \frac{\overset{\text{Likelihood}}{P(x \mid c)}\,\overset{\text{Class Prior Probability}}{P(c)}}{\underset{\text{Predictor Prior Probability}}{P(x)}}$$

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

Above,

- *P*(*c*|*x*) is the posterior probability of *class* (c, *target*) given *predictor* (x, *attributes*).
- *P*(*c*) is the prior probability of *class*.
- *P*(*x*|*c*) is the likelihood which is the probability of *predictor* given *class*.
- *P*(*x*) is the prior probability of *predictor*.

How Naive Bayes algorithm works?

Let's understand it using an example. Below I have a training data set of weather and corresponding target variable 'Play' (suggesting possibilities of playing). Now, we need to classify whether players will play or not based on weather condition. Let's follow the below steps to perform it.

Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

Problem: Players will play if weather is sunny. Is this statement is correct?

We can solve it using above discussed method of posterior probability.

P(Yes | Sunny) = P(Sunny | Yes) * P(Yes) / P (Sunny)

Here we have P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P(Yes)= 9/14 = 0.64

Now, P (Yes | Sunny) = 0.33 * 0.64 / 0.36 = 0.60, which has higher probability.

Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

What are the Pros and Cons of Naive Bayes?

Pros:

- It is easy and fast to predict class of test data set. It also perform well in multi class prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Cons:

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as “Zero Frequency”. To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
- On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict_proba are not to be taken too seriously.
- Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

4 Applications of Naive Bayes Algorithms

- Real time Prediction: Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in real time.
- Multi class Prediction: This algorithm is also well known for multi class prediction feature. Here we can predict the probability of multiple classes of target variable.
- Text classification/ Spam Filtering/ Sentiment Analysis: Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)
- Recommendation System: Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not

How to build a basic model using Naive Bayes in Python and R?

Again, scikit learn (python library) will help here to build a Naive Bayes model in Python. There are three types of Naive Bayes model under the scikit-learn library:

- Gaussian: It is used in classification and it assumes that features follow a normal distribution.
- Multinomial: It is used for discrete counts. For example, let’s say, we have a text classification problem. Here we can consider Bernoulli trials which is one step further and instead of “word occurring in the document”, we have “count how often word occurs in the document”, you can think of it as “number of times outcome number x_i is observed over the n trials”.
- Bernoulli: The binomial model is useful if your feature vectors are binary (i.e. zeros and ones). One application would be text classification with ‘bag of words’ model where the 1s & 0s are “word occurs in the document” and “word does not occur in the document” respectively.

……………………………………………………………………………..

3. k-NN algorithm

Introduction

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry. The following two properties would define KNN well −

- Lazy learning algorithm − KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification.
- Non-parametric learning algorithm − KNN is also a non-parametric learning algorithm because it doesn't assume anything about the underlying data.

Working of KNN Algorithm

K-nearest neighbors (KNN) algorithm uses 'feature similarity' to predict the values of new data points which further means that the new data point will be assigned a value based on how closely it matches the points in the training set. We can understand its working with the help of following steps −

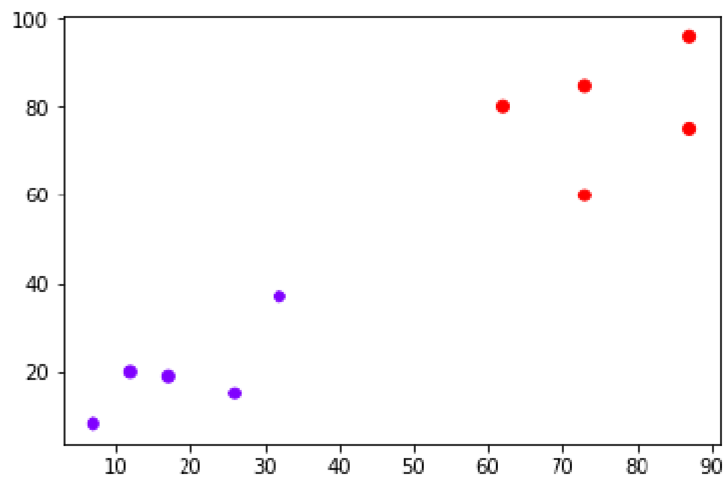
Step 1 − For implementing any algorithm, we need dataset. So during the first step of KNN, we must load the training as well as test data.

Step 2 − Next, we need to choose the value of K i.e. the nearest data points. K can be any integer.

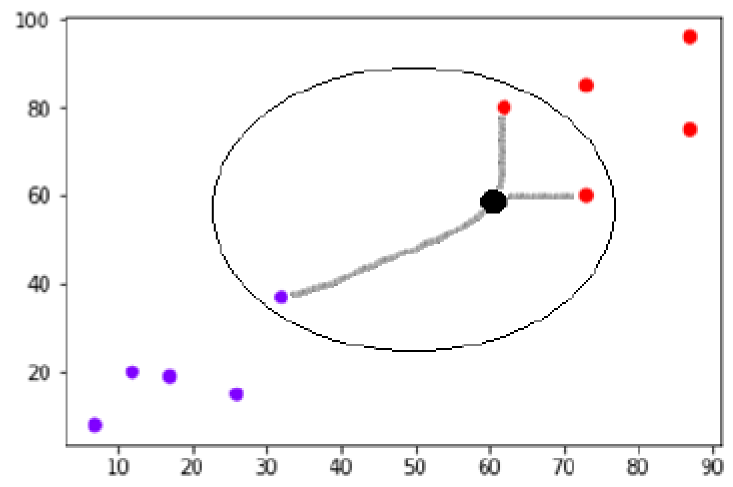
Step 3 − For each point in the test data do the following −

- 3.1 − Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.
- 3.2 − Now, based on the distance value, sort them in ascending order.
- 3.3 − Next, it will choose the top K rows from the sorted array.
- 3.4 − Now, it will assign a class to the test point based on most frequent class of these rows.

Step 4 − End



Now, we need to classify new data point with black dot (at point 60,60) into blue or red class. We are assuming K = 3 i.e. it would find three nearest data points. It is shown in the next diagram −



We can see in the above diagram the three nearest neighbors of the data point with black dot. Among those three, two of them lies in Red class hence the black dot will also be assigned in red class.

Implementation in Python

As we know K-nearest neighbors (KNN) algorithm can be used for both classification as well as regression. The following are the recipes in Python to use KNN as classifier as well as regressor −

KNN as Classifier

First, start with importing necessary python packages −

import numpy as np

```
import matplotlib. pyplot as plt
import pandas as pd
```

Next, download the iris dataset from its web link as follows −

```
path = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
```

Next, we need to assign column names to the dataset as follows −

```
header names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']
```

Now, we need to read dataset to pandas data frame as follows −

```
dataset = pd. read_csv (path, names = header names)
dataset. Head ()
```

	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Data Preprocessing will be done with the help of following script lines.

```
X = dataset. iloc[:, :-1].values
y = dataset. iloc[:, 4].values
```

Next, we will divide the data into train and test split. Following code will split the dataset into 60% training data and 40% of testing data −

```
from sklearn.model_selection import train_test_split
X_train, Test, train, y_test = train_test_split (X, y, test size = 0.40)
```

Next, data scaling will be done as follows −

```
from sklearn. preprocessing import StandardScaler
scaler = StandardScaler ()
scaler.fit(X_train)
X_train = scaler. transform(X_train)
X_test = scaler. transform(X_test)
```

Next, train the model with the help of KNeighborsClassifier class of sklearn as follows −

```
from sklearn. Neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier (neighbors = 8)
classifier. Fit (X_train, y_train)
```

At last we need to make prediction. It can be done with the help of following script −

```
y_pred = classifier. Predict(X_test)
```

Next, print the results as follows −

```
from sklearn. Metrics import classification report, confusion matrix, accuracy score
result = confusion matrix (y_test, y_pred)
print ("Confusion Matrix:")
print(result)
result1 = classification report (y_test, y_pred)
print ("Classification Report:",)
print (result1)
result2 = accuracy score (y_test, y_pred)
print ("Accuracy:”, result2)
```

Output

```
Confusion Matrix:
[[21 0 0]
[ 0 16 0]
[ 0 7 16]]
Classification Report:
               precision   recall   f1-score   support
   Iris-setosa       1.00     1.00       1.00       21
Iris-versicolor       0.70     1.00       0.82       16
 Iris-Virginia       1.00     0.70       0.82       23
     micro avg       0.88     0.88       0.88       60
     macro avg       0.90     0.90       0.88       60
  weighted avg       0.92     0.88       0.88       60

Accuracy: 0.8833333333333333
```

KNN as Regressor

First, start with importing necessary Python packages −

```
import numpy as np
import pandas as pd
```

Next, download the iris dataset from its web link as follows −

```
path = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
```

Next, we need to assign column names to the dataset as follows −

```
header names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']
```

Now, we need to read dataset to pandas data frame as follows −

```
data = pd. read_csv(url, names = header names)
array = data. Values
X = array[:,:2]
Y = array[:,2]
data. Shape
output:(150, 5)
```

Next, import *KNeighborsRegressor* from *sklearn* to fit the model −

```
from sklearn. Neighbors import KNeighborsRegressor
knnr = KNeighborsRegressor (neighbors = 10)
knnr. Fit(X, y)
```

At last, we can find the MSE as follows −

```
print ("The MSE is:", format (np.power(y-knnr. Predict(X),2). mean()))
Output
The MSE is: 0.12226666666666669
```

Pros and Cons of KNN

Pros

- It is very simple algorithm to understand and interpret.
- It is very useful for nonlinear data because there is no assumption about data in this algorithm.
- It is a versatile algorithm as we can use it for classification as well as regression.
- It has relatively high accuracy but there are much better supervised learning models than KNN.

Cons

- It is computationally a bit expensive algorithm because it stores all the training data.
- High memory storage required as compared to other supervised learning algorithms.
- Prediction is slow in case of big N.
- It is very sensitive to the scale of data as well as irrelevant features.

Applications of KNN

The following are some of the areas in which KNN can be applied successfully −

Banking System

KNN can be used in banking system to predict weather an individual is fit for loan approval? Does that individual have the characteristics similar to the defaulters one?

Calculating Credit Ratings

KNN algorithms can be used to find an individual's credit rating by comparing with the persons having similar traits.

Politics

With the help of KNN algorithms, we can classify a potential voter into various classes like "Will Vote", "Will not Vote", "Will Vote to Party 'Congress', "Will Vote to Party 'BJP'.

Other areas in which KNN algorithm can be used are Speech Recognition, Handwriting Detection, Image Recognition and Video Recognition.

When do we use KNN algorithm?

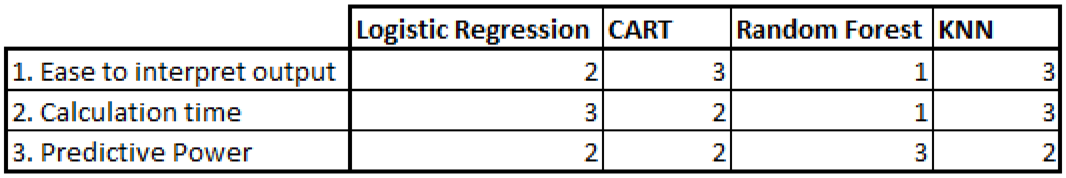
KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique, we generally look at 3 important aspects:

1. Ease to interpret output

2. Calculation time

3. Predictive Power

Let us take a few examples to place KNN in the scale:



	Logistic Regression	CART	Random Forest	KNN
1. Ease to interpret output	2	3	1	3
2. Calculation time	3	2	1	3
3. Predictive Power	2	2	3	2

KNN algorithm fairs across all parameters of considerations. It is commonly used for its easy of interpretation and low calculation time.

………………………… END………………………..