Exercise 4: Classification

Breast Cancer Data

1. (a) The class distribution is binary.

   (b) Categoric: menopause, breast* (or binary) , breast-quad;

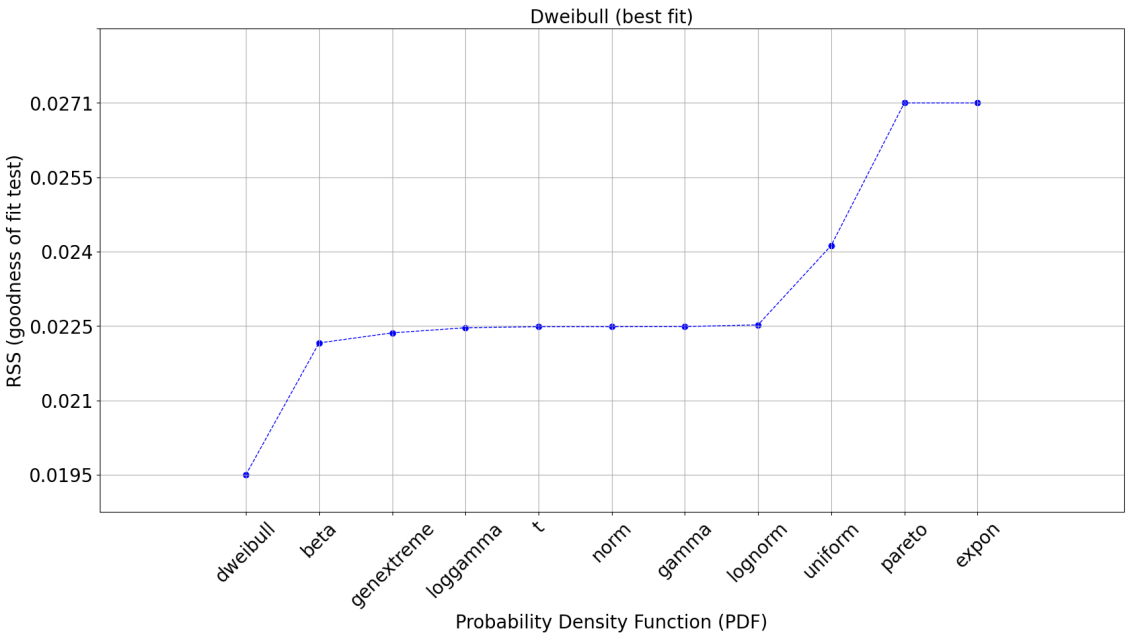
   Ordered: age, tumor-size, inv-nodes, deg-malig;

   Binary: node-caps, irradiat.

   (c) We use $(a+b)/2$ to "represent" each age group $[a–b]$

```
X = np.array([24.5] * 1 + [34.5] * 36 + [44.5] * 90 \
    + [54.5] * 96 + [64.5] * 57 + [74.5] * 6)

dfit = distfit()
results = dfit.fit_transform(X)
dfit.plot_summary()
```



We choose normal distribution because it is simply and its RRS is... okay. Well there must be some loss of precision since everything between a and b is treated as $(a+b)/2$.

```
dfit = distfit(distr='norm')
dfit.fit_transform(X)
print(dfit.model)
```

For this normal distribution $\mu = 51.1434$ and $\sigma = 10.1005$.

(d) Apply Weka's CorrelationAttributeEval, "node-caps" show the most dependence on the class value as it comes first in the Correlation Rank. (The order of all attributes is node-caps, inv-nodes, deg-malig, irradiat, tumor-size, breast, menopause, breast-quad, age.)

2. (a) The accuracy percentage of the majority class classifier (Zeror) is 65.9794%.

   The model predicts "no-recurrence-events" for every instance it is given.

   (b) The accuracy of the NaiveBayes classifier is 71.134%.

   (c) $k \in \{4, 5, 6, 7\}$ for KNN gives the best accuracy on this data.

k	*1*	*2*	*3*	*4, 5, 6, 7*	*8*	*9, 10*
accuracy	72.1649%	69.0722%	70.1031%	73.1959%	72.1649%	70.1031%

   (d) The results of running the nearest neighbour classifier with Manhattan Distance and Euclidean Distance are the same for each k in KNN, because all attributes in the dataset are (represented and interpreted as) nominal rather than numeric in Weka.

   Looking up the source code for calculating distance, we find:

```
public abstract class NormalizableDistance
  ......
  protected double difference(int index, double val1, double val2) {
    switch (m_Data.attribute(index).type()) {
    case Attribute.NOMINAL:
      if (Utils.isMissingValue(val1) || Utils.isMissingValue(val2)
        || ((int) val1 != (int) val2)) {
        return 1;
      } else {
        return 0;
      }
      ......
```

   We understand that all different values for a nominal attribute are "equally different". Intuitively, we would expect the difference between, for example, age $[10–19]$ and $[90–99]$ to be much greater than that between $[10–19]$ and $[20–29]$, but in fact they would both be 1. For any two instances, if they have n attributes with different values, the Euclidean distance would be $\sqrt{n}$ and the Manhattan distance would be n, thus we get exactly the same rank order as well as the same "nearest neighbours" using both distance functions.

   (e) The accuracy of the J48 classifier with default parameters is 68.0412%.

   The decision tree first split into $node\text{–}caps = yes$ and $node\text{–}caps = no$ from root. For $node\text{–}caps = no$ it simply outputs no-recurrence-events as class; for $node\text{–}caps = yes$ it then checks $deg\text{–}malig$, if $deg\text{–}malig$ is 2 it outputs no-recurrence-events, otherwise for 1 or 3 it outputs recurrence-events. The tree has 6 nodes and 4 leaves.

```
J48 pruned tree
------------------
node-caps = yes
|   deg-malig = 1: recurrence-events (1.01/0.4)
|   deg-malig = 2: no-recurrence-events (26.2/8.0)
|   deg-malig = 3: recurrence-events (30.4/7.4)
node-caps = no: no-recurrence-events (228.39/53.4)

=== Summary ===
Correctly Classified Instances          66               68.0412 %
Incorrectly Classified Instances        31               31.9588 %
Total Number of Instances               97
```

From this we can tell that among all 286 instances in the entire dataset, 69.2 of them are missclassified (here we have fractions because the dataset has missing values). Among 97 instances in the test set, 66 are classified correctly. Thus accuracy of this classifier on training set can be calculated: $(286 - 69.2 - 66)/(286 - 97) = 79.79\%$. The model's performance on the training dataset is a lot better than on the test dataset, so it may be overfitting.

(f) "deg-malig=3" is the most heavily weighted, followed by "node-caps=no" and "inv-nodes=0-2". Recall the decision tree classifier, deg-malig and node-caps were also used when the data is divided into branches. These are the two most effective attributes to predict the class.

3. The Naive Bayes classifier has both the lowest False Positive rate (51.5%) and the highest precision (75.7%) for the no-recurrence class. For the application of predicting recurrence of breast cancer, the FP rate means for 51.5% of the cases where there actually is a recurrence, the model wrongly classified them as no recurrence, which is very serious as these patients may then miss the best period of treatment, it is not acceptable. As for the precision, it means 75.7% of the cases predicted as no recurrence was correct while the others were wrong, also not very acceptable, as we would like to find out the patients that are in danger of a recurrence as many as possible, and it would be terrible to miss any of them.

Car Data

4. (a) The number of examples for each value of an attribute are always the same, suppose the dataset has N instances, for each attribute with x different values, each value occur exactly N/x times. It is also observed that the product of all the attributes' x is N, so every combination of the attribute values exists and appears only once in the dataset (we may view the dataset as a permutation of the attributes possible values). The data is collected that way.

(b) The SVM classifier gives the best accuracy, followed by J48, KNN and Naive Bayes. For KNN all K's from 1 to 6 yields the same accuracy, increasing K will not make the accuracy increase, on the contrary, it becomes lower as K becomes larger. This is probably due to the attribute values are so evenly distributed with no duplication in any sense.

model	***ZeroR***	***NaiveBayes***	***J48***	***SVM***
accuracy	69.5578%	87.585%	90.9864%	93.3673%

KNN	k = 1, 2, 3, 4, 5, 6	k = 7, 8	k = 9	k = 10
accuracy	90.6463%	90.1361%	87.7551%	85.8844%

5. (a) Cars with low safety and/or can only carry 2 people are unacceptable.

   (b) The accuracy for the J48 decision tree is 94.2177% and for the SVM model it's 94.3878%. The decision tree is generally more interpretable to humans than the SVM (both trees for multivalued class variable and binary class variable are easy to understand). As we browse down the tree we always see the leaves of a node going from unacc to acc (for binary class, then good to vgood for multivalued class), so we realise that for safety it's the higher the better, for capacity it is the larger the better, for buying and maintenance price the lower the better, the size of luggage boot the bigger the better, and the number of doors the more the better. The model aims to seek a balance or compromise among these factors to evaluate whether a car is worth buying or if so, just how good a deal it is.

   The SVM model on binary class is more interpretable than that on multivalued class. If an attribute's weight is negative, we know it is a disadvantage that makes a car less "buyable", for example when buying price is "very high"; if the weight is positive, it is an advantage that makes a car more worth buying, for example when a car has 4 doors. And the absolute value of the weight measures how much it affects the final decision. For example, the weight for "3 doors" and "4 doors" are both positive, but the absolute value of "4 doors" is greater, so the more doors, the better, the same idea we get from the decision tree, also quite intuitive. However for the multivalued class SVM, it actually builds 6 (C_4^2) binary between each pair of values, so it is more complicated (redundant) and less interpretable to the eye.

Problem Set 4

1. (a) $a = 0$, $Y = \varepsilon$ and $\varepsilon \sim Ber(p)$, apply the Maximum Likelihood Estimator:

   $L(p) = \prod_{i=1}^{n} p^{y_i}(1-p)^{1-y_i} = p^S(1-p)^{n-S},\ S := \sum_{i=0}^{n} y_i$

   $l(p) = S\log(p) + (n-S)\log(1-p),$ set $\frac{\partial l(p)}{\partial p} = 0$

   $\hat{p} = \frac{S}{n} = \frac{1}{n}\sum_{i=0}^{n} y_i$

   (b) $Y - aX = \varepsilon$ and $\varepsilon \sim Ber(p)$, apply the Maximum Likelihood Estimator:

   $L(p,a) = \prod_{i=1}^{n} p^{y_i - ax_i}(1-p)^{1-(y_i-ax_i)} = p^{S-aT}(1-p)^{n-(S-aT)}$

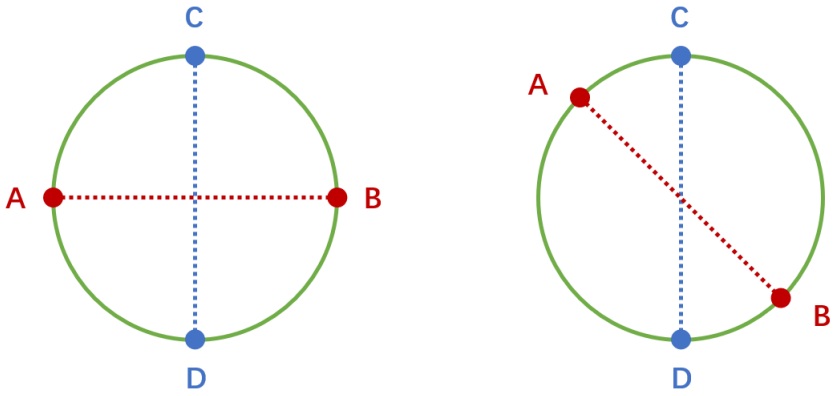
   $S := \sum_{i=0}^{n} y_i,\ T := \sum_{i=0}^{n} x_i$

   $l(p,a) = (S-aT)\log(p) + (n-S+aT)\log(1-p)$

   set $\frac{\partial l(p)}{\partial p} = 0$ and $\frac{\partial l(p)}{\partial a} = 0$

   $\hat{p} = \frac{1}{2},\ \hat{a} = \frac{2S-n}{2T} = (2\sum_{i=0}^{n} y_i - n)/2\sum_{i=0}^{n} x_i$

2. (a) Draw a diameter of the circle, place A and B on the two ends, draw another diameter (could be any random one as long as the two are not exactly the same), place C and D on its two ends. Try 4-fold cross validation on KNN classifier with $k = 2$, for train set $\{A, B, C\}$ and test set $\{D\}$, D will be classified according to the class of A and B, which is wrong, and so on. For all 4 "folds" the test point would be classified wrongly, so the error is 1.



(b) Assume for points on the same circle we have the normal Euclidean distance as in (a), then A and B will always have each other in their 2 nearest neighbours, as $d(A, B) \leq 1$, $d(B, A) \leq 1$ while $d(x, y) > 1$ for $x \in \{A, B\}$ and $y \in \{C, D\}$. As for C and D, their 2 nearest neighbours will always be A and B, as we have $d(C, D) \geq 0$, $d(D, C) \geq 0$ and $d'(y, x) = -d(x, y) < -1$ for $x \in \{A, B\}$ and $y \in \{C, D\}$.

We must specify how to break ties: apparently C, D must choose between A and B (which one to go with). If we just pick one of the 2 nearest neighbours at random, then it will always be possible for the 4-fold cross validation error to be 1: to classify A we pick B, for B we pick A, for C we pick B and for D we pick A, all of them are wrong, the error is 1.

On the other hand, if we decide to always choose the one "nearer" neighbour out of the 2 nearest when they happen to have different class labels, then A and B will be classified according to each other, wrong. In order for C and D to be classified wrongly as well, we would like $d(C, B) < d(C, A)$ and $d(D, A) < d(D, B)$, that is to say, $d(B, C) > d(A, C)$ and $d(A, D) > d(B, D)$. A possible case to yield error 1 is shown below.