Homework 2

Your Name: **Srujan Shekar Shetty**

Student ID: **A20529733**

---

1. (50 points) Classification: We are going to make a decision about whether an animal is useful (P) or useless (N) in our experiments. We measure their age in days, whether fat or not, and the size of their soles of the feet. [Manually solve this problem without Python]

Left Back	Right Back	Left Front	Right Front	Fat	Age	Label
4.1	4.8	1.6	3.2	Yes	100	P
4.6	4.2	1.4	0.2	Yes	40	N
4.3	5.0	1.5	4.2	No	160	N
5	1.3	1.4	2.2	Yes	90	N
5	1.2	4.7	1.4	No	40	N
4.4	3.2	4.5	1.5	No	80	P
4.9	3.1	4.9	1.5	No	100	N
2.5	1.3	4	1.3	Yes	110	P
4.5	2.8	4.6	1.5	Yes	120	P
4.3	3.3	4.9	2.5	Yes	30	P
1.8	2.7	5.0	1.9	Yes	20	P
2.1	3	5.0	2.1	No	40	N
4.3	2.9	5.0	1.8	No	30	P
4.5	3	4.9	2.2	No	50	N
4.3	**3.6**	**3.8**	**1.9**	**Yes**	**70**	**?**

a). [10 points] Do we need normalization and discretization (data type transformation) to use KNN classifier? Why (use your own text/description)?
Ans-

- We know that Normalization and Discretization are preprocessing techniques that are under Data Transformation. Yes, We need Normalization and Discretization.
- Data transformation is needed to avoid difficulties when:
  a) Attribute values are at different scales meaning if two different data sets measure distance in Km and Meters there shall be a problem in merging the data set to analyze it.
  b) Different data formats which can lead to difficulties in data comparison.

- Specifically with respect to the KNN classifier. It is recommended to go with the data preprocessing technique of Data Normalization and Data discretization.
- Data Normalization can produce values in a new and precise scale without taking away the meaning of it which can avoid large number calculations causing bias when we use the KNN classifier Manhattan distance, Euclidean distance, or any other distance calculation method.
- Data Discretization does the conversion of nominal data to numeric data which can help to measure the Manhattan distance or any other distance calculation method that is used in the KNN classification.
- In our problem, we can use Data Normalization and Data discretization to fix a standard scale and convert the Fat feature from nominal to numerical to help us calculate the Manhattan distance.
- Therefore we can conclude that Normalization is necessary; else, the computation of distance will be affected by larger features.

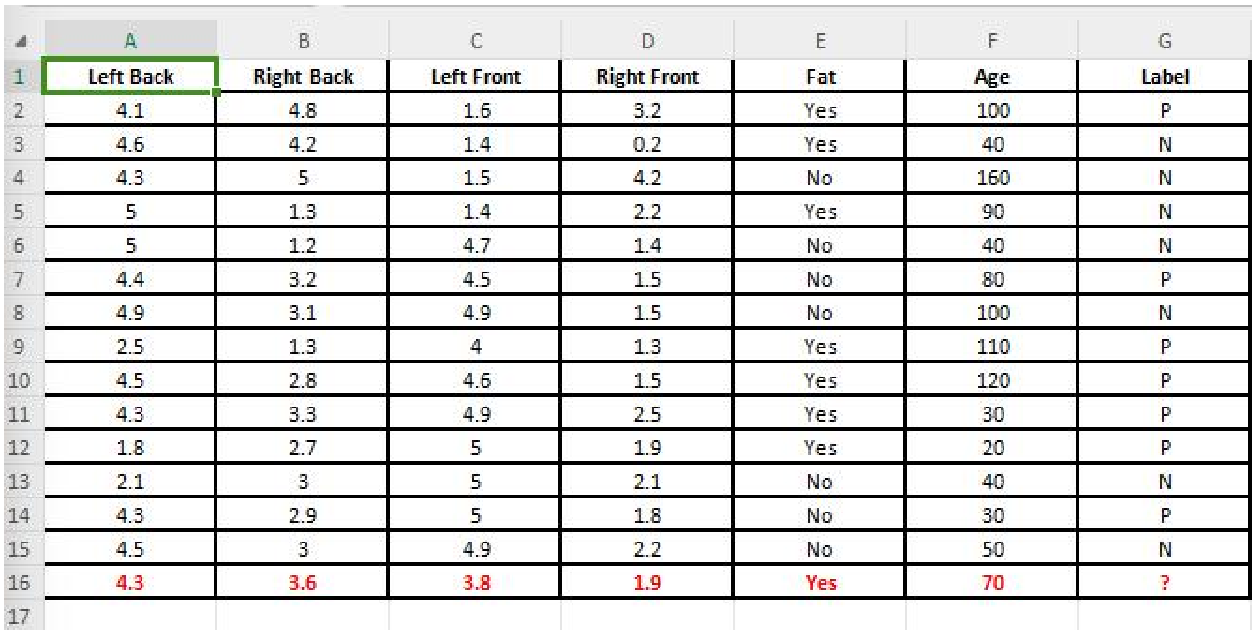
b). [10 points] If your answer is Yes in part 1), please apply normalization (to new scale [1,5]) and discretization. Give the process of preprocessing and the table of final data

Ans-

<u>Applying Normalization:</u> Here in our case, we shall apply Min-Max Normalization to the data set.
We know that,

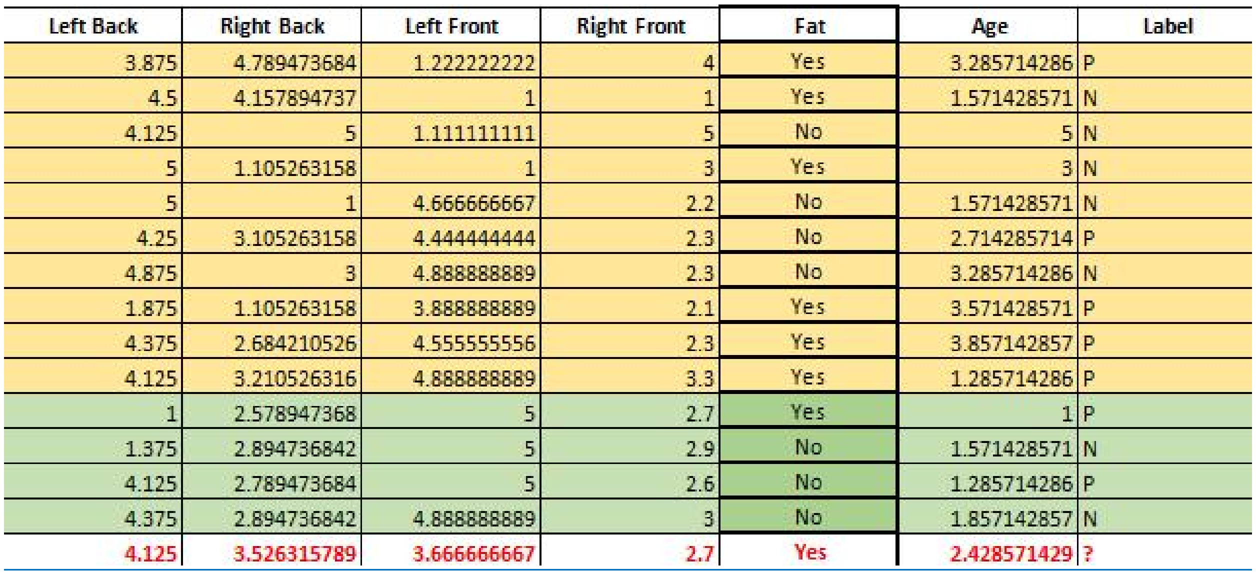
$$\boxed{x'_i = \frac{x_i - \min x_i}{\max x_i - \min x_i}(new\max - new\min) + new\min}$$

Step 1: Copy the dataset to Excel.



	A	B	C	D	E	F	G
1	Left Back	Right Back	Left Front	Right Front	Fat	Age	Label
2	4.1	4.8	1.6	3.2	Yes	100	P
3	4.6	4.2	1.4	0.2	Yes	40	N
4	4.3	5	1.5	4.2	No	160	N
5	5	1.3	1.4	2.2	Yes	90	N
6	5	1.2	4.7	1.4	No	40	N
7	4.4	3.2	4.5	1.5	No	80	P
8	4.9	3.1	4.9	1.5	No	100	N
9	2.5	1.3	4	1.3	Yes	110	P
10	4.5	2.8	4.6	1.5	Yes	120	P
11	4.3	3.3	4.9	2.5	Yes	30	P
12	1.8	2.7	5	1.9	Yes	20	P
13	2.1	3	5	2.1	No	40	N
14	4.3	2.9	5	1.8	No	30	P
15	4.5	3	4.9	2.2	No	50	N
16	4.3	3.6	3.8	1.9	Yes	70	?
17							

Step 2: Apply the above formula to all the features except Fat (Since we shall use discretization for it).



Left Back	Right Back	Left Front	Right Front	Fat	Age	Label
3.875	4.789473684	1.222222222	4	Yes	3.285714286	P
4.5	4.157894737	1	1	Yes	1.571428571	N
4.125	5	1.111111111	5	No	5	N
5	1.105263158	1	3	Yes	3	N
5	1	4.666666667	2.2	No	1.571428571	N
4.25	3.105263158	4.444444444	2.3	No	2.714285714	P
4.875	3	4.888888889	2.3	No	3.285714286	N
1.875	1.105263158	3.888888889	2.1	Yes	3.571428571	P
4.375	2.684210526	4.555555556	2.3	Yes	3.857142857	P
4.125	3.210526316	4.888888889	3.3	Yes	1.285714286	P
1	2.578947368	5	2.7	Yes	1	P
1.375	2.894736842	5	2.9	No	1.571428571	N
4.125	2.789473684	5	2.6	No	1.285714286	P
4.375	2.894736842	4.888888889	3	No	1.857142857	N
4.125	3.526315789	3.666666667	2.7	Yes	2.428571429	?

The above table shows data after Normalization.

Applying Discretization to FAT feature:

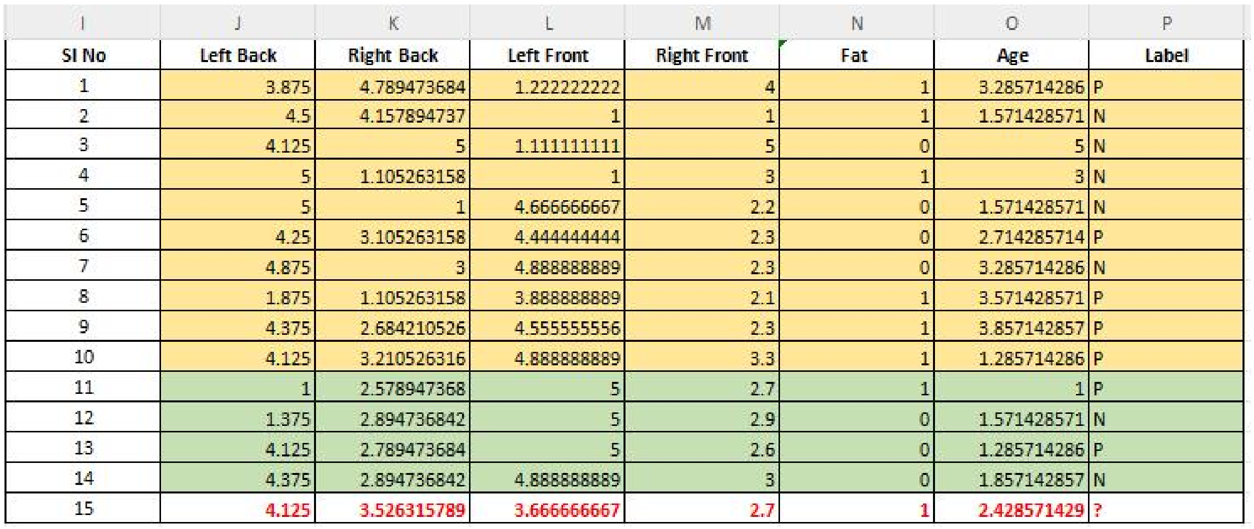
Conversion of nominal data to numeric data which shall help to measure the Manhattan distance or any other type of distance.

Assume there are N values in a variable, you just need to create N-1 new columns. In our case, it is Yes and No. So, we need to create only one column in fact replace Yes with 1 and No with 0

Step 1: Copy the dataset to Excel.

Step 2: Convert the nominal value to a numeric value (Binary in this case)

We can manually convert the value or use the Excel formula as follows to do it:
IF (N2= “Yes”,1,0)
In the above formula, N2 represents the value in the column name FAT.



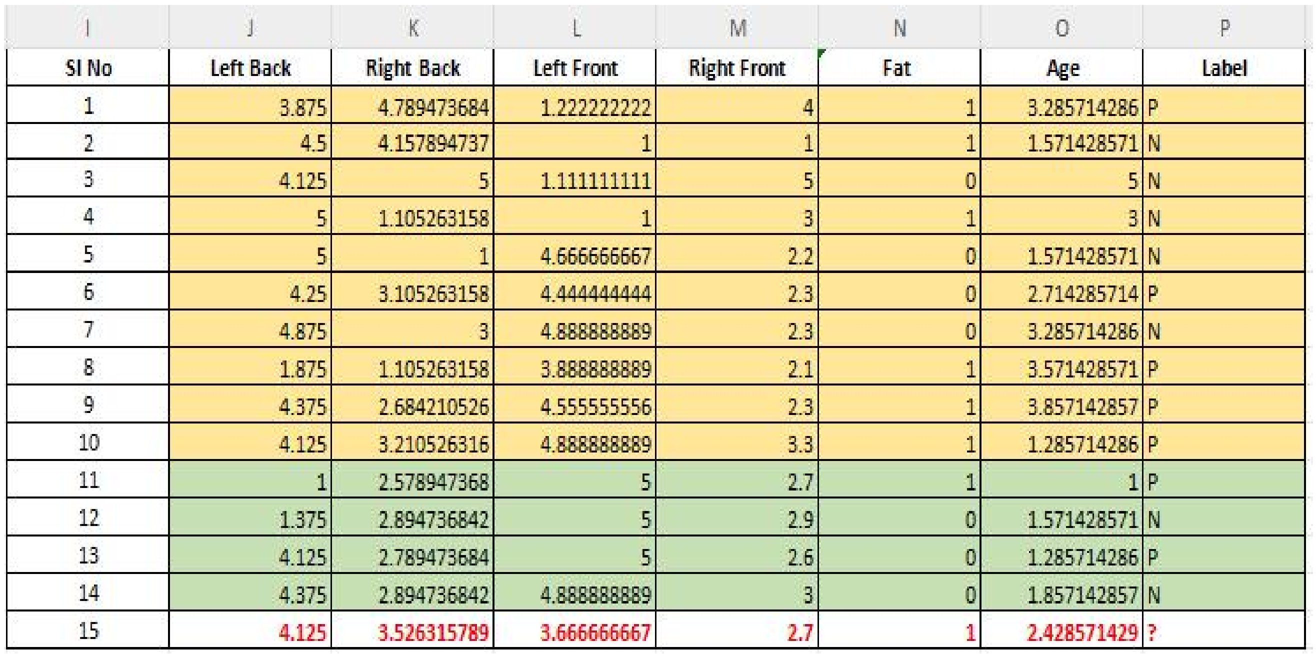
I	J	K	L	M	N	O	P
Sl No	Left Back	Right Back	Left Front	Right Front	Fat	Age	Label
1	3.875	4.789473684	1.222222222	4	1	3.285714286	P
2	4.5	4.157894737	1	1	1	1.571428571	N
3	4.125	5	1.111111111	5	0	5	N
4	5	1.105263158	1	3	1	3	N
5	5	1	4.666666667	2.2	0	1.571428571	N
6	4.25	3.105263158	4.444444444	2.3	0	2.714285714	P
7	4.875	3	4.888888889	2.3	0	3.285714286	N
8	1.875	1.105263158	3.888888889	2.1	1	3.571428571	P
9	4.375	2.684210526	4.555555556	2.3	1	3.857142857	P
10	4.125	3.210526316	4.888888889	3.3	1	1.285714286	P
11	1	2.578947368	5	2.7	1	1	P
12	1.375	2.894736842	5	2.9	0	1.571428571	N
13	4.125	2.789473684	5	2.6	0	1.285714286	P
14	4.375	2.894736842	4.888888889	3	0	1.857142857	N
15	4.125	3.526315789	3.666666667	2.7	1	2.428571429	?

The above screenshot represents the data after discretization.

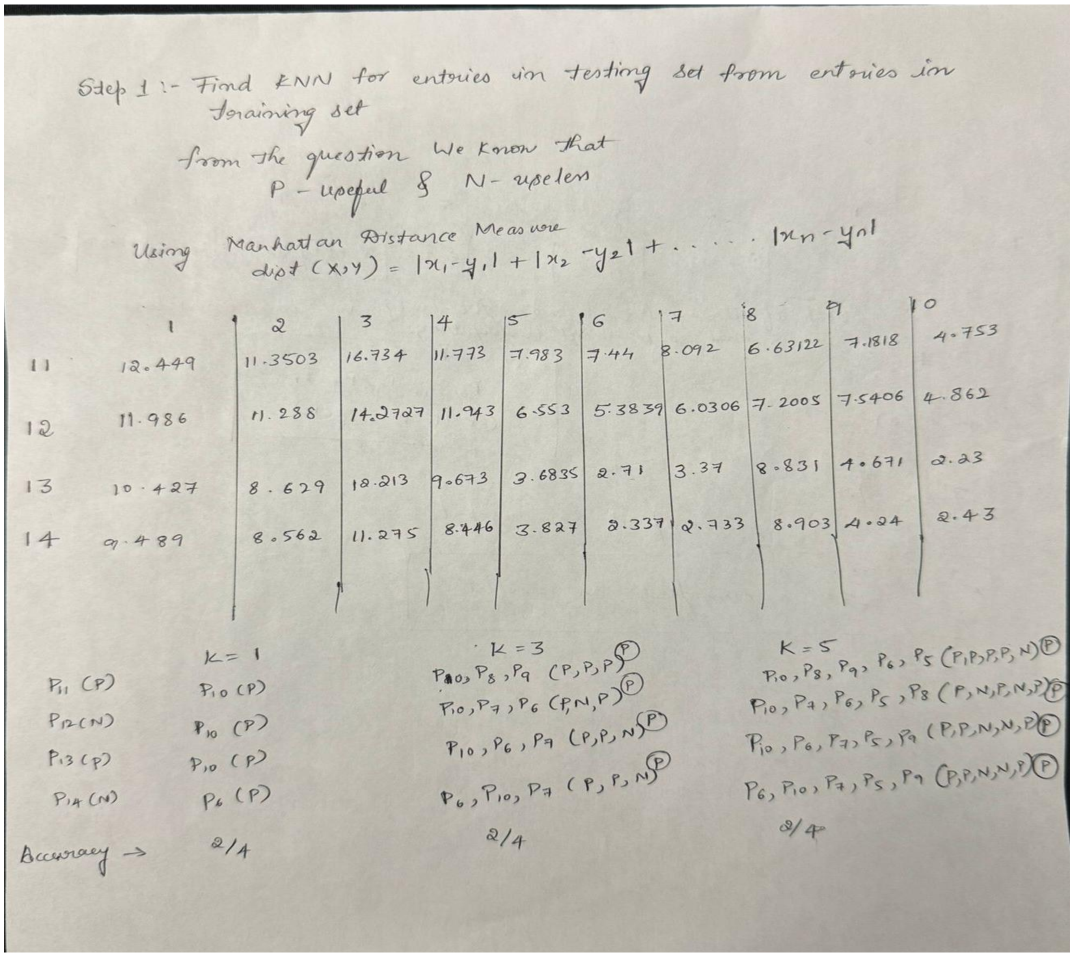
c). [30 points] Use first 10 rows as training, the next 4 rows as testing set. Apply KNN Classifier to the new data table in part b). In other words, build your KNN classifier by the following requirements based on the knowledge in the table, and then predict which class/label the object (in red) belongs to:

- Distance measures: Manhattan distance
- K = 1, 3, 5

Find the best K value by examining accuracy on the test set. Finally apply the best setting and predict the label for the unseen data.



I	J	K	L	M	N	O	P
SI No	Left Back	Right Back	Left Front	Right Front	Fat	Age	Label
1	3.875	4.789473684	1.222222222	4	1	3.285714286	P
2	4.5	4.157894737	1	1	1	1.571428571	N
3	4.125	5	1.111111111	5	0	5	N
4	5	1.105263158	1	3	1	3	N
5	5	1	4.666666667	2.2	0	1.571428571	N
6	4.25	3.105263158	4.444444444	2.3	0	2.714285714	P
7	4.875	3	4.888888889	2.3	0	3.285714286	N
8	1.875	1.105263158	3.888888889	2.1	1	3.571428571	P
9	4.375	2.684210526	4.555555556	2.3	1	3.857142857	P
10	4.125	3.210526316	4.888888889	3.3	1	1.285714286	P
11	1	2.578947368	5	2.7	1	1	P
12	1.375	2.894736842	5	2.9	0	1.571428571	N
13	4.125	2.789473684	5	2.6	0	1.285714286	P
14	4.375	2.894736842	4.888888889	3	0	1.857142857	N
15	4.125	3.526315789	3.666666667	2.7	1	2.428571429	?



Step 1 :- Find KNN for entries in testing set from entries in training set

from the question We know that

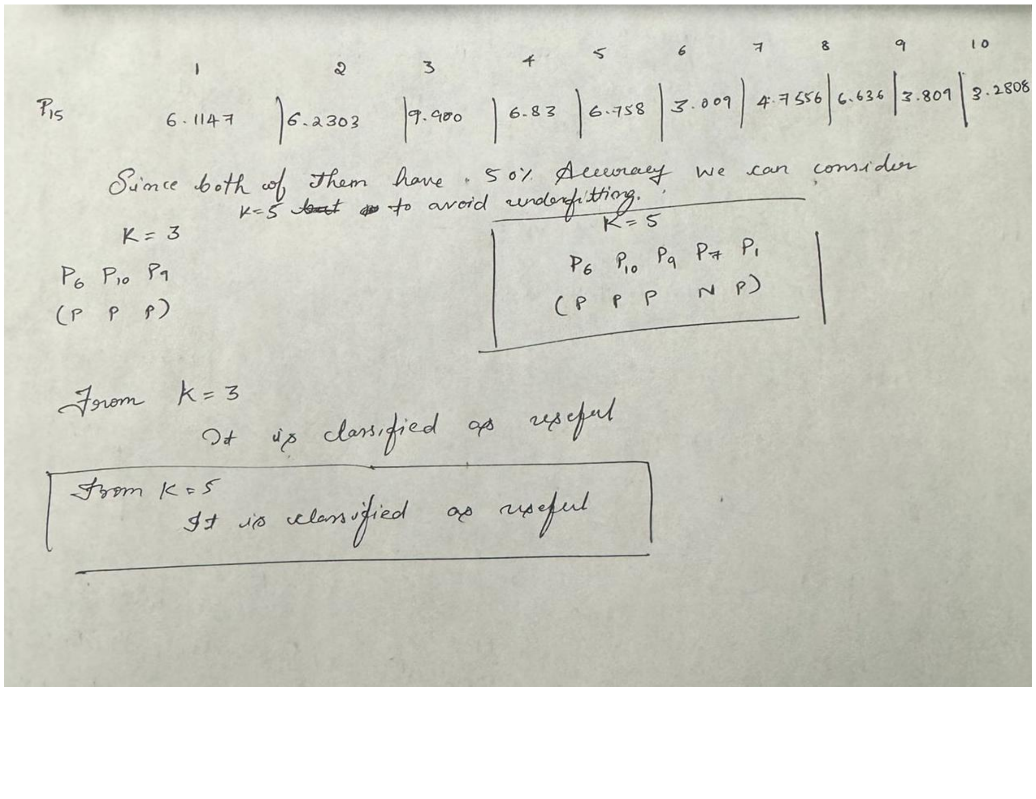
P - useful & N - useless

Using Manhattan Distance Measure

$dist(x,y) = |x_1 - y_1| + |x_2 - y_2| + \dots |x_n - y_n|$

	1	2	3	4	5	6	7	8	9	10
11	12.449	11.3503	16.734	11.773	7.983	7.44	8.092	6.63122	7.1818	4.753
12	11.986	11.288	14.2727	11.943	6.553	5.3839	6.0306	7.2005	7.5406	4.862
13	10.427	8.629	12.213	9.673	3.6835	2.71	3.37	8.831	4.671	2.23
14	9.489	8.562	11.275	8.446	3.827	2.337	2.733	8.903	4.24	2.43

	K = 1	K = 3	K = 5
P_{11} (P)	P_{10} (P)	P_{10}, P_8, P_9 (P, P, P) Ⓟ	$P_{10}, P_8, P_9, P_6, P_5$ (P, P, P, P, N) Ⓟ
P_{12} (N)	P_{10} (P)	P_{10}, P_7, P_6 (P, N, P) Ⓟ	$P_{10}, P_7, P_6, P_5, P_8$ (P, N, P, N, P) Ⓟ
P_{13} (P)	P_{10} (P)	P_{10}, P_6, P_7 (P, P, N) Ⓟ	$P_{10}, P_6, P_7, P_5, P_9$ (P, P, N, N, P) Ⓟ
P_{14} (N)	P_6 (P)	P_6, P_{10}, P_7 (P, P, N) Ⓟ	$P_6, P_{10}, P_7, P_5, P_9$ (P, P, N, N, P) Ⓟ
Accuracy →	2/4	2/4	2/4



	1	2	3	4	5	6	7	8	9	10
P_{15}	6.1147	6.2303	9.900	6.83	6.758	3.009	4.7556	6.636	3.809	3.2808

Since both of them have 50% Accuracy we can consider K=5 ~~but~~ to avoid underfitting.

K = 3

P_6 P_{10} P_9

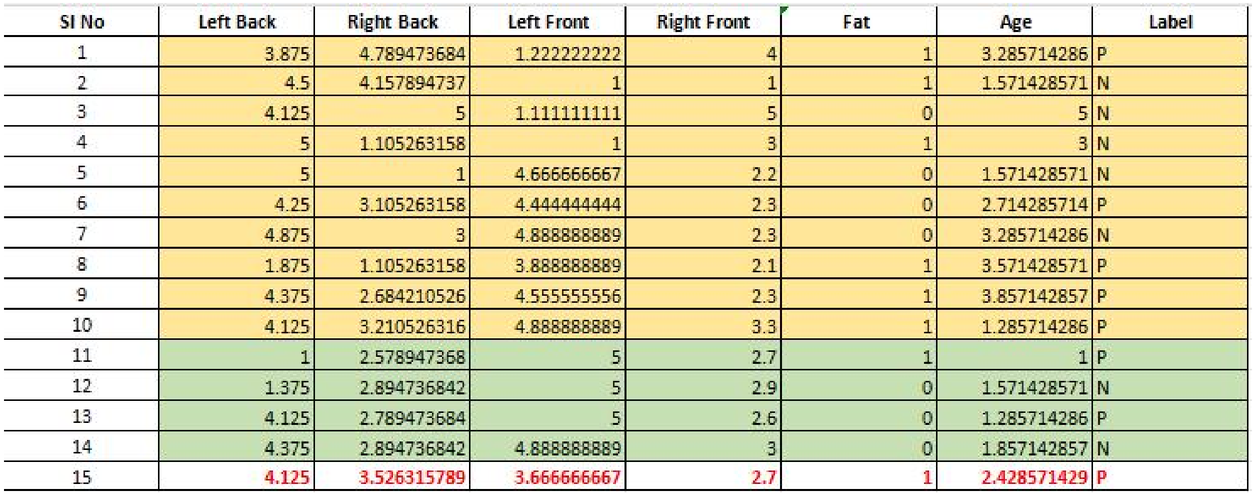
(P P P)

$$\boxed{\begin{array}{ccccc} & & K=5 & & \\ P_6 & P_{10} & P_9 & P_7 & P_1 \\ (P & P & P & N & P) \end{array}}$$

From K = 3

It is classified as useful

$$\boxed{\text{From } K=5 \quad \text{It is classified as useful}}$$



Sl No	Left Back	Right Back	Left Front	Right Front	Fat	Age	Label
1	3.875	4.789473684	1.222222222	4	1	3.285714286	P
2	4.5	4.157894737	1	1	1	1.571428571	N
3	4.125	5	1.111111111	5	0	5	N
4	5	1.105263158	1	3	1	3	N
5	5	1	4.666666667	2.2	0	1.571428571	N
6	4.25	3.105263158	4.444444444	2.3	0	2.714285714	P
7	4.875	3	4.888888889	2.3	0	3.285714286	N
8	1.875	1.105263158	3.888888889	2.1	1	3.571428571	P
9	4.375	2.684210526	4.555555556	2.3	1	3.857142857	P
10	4.125	3.210526316	4.888888889	3.3	1	1.285714286	P
11	1	2.578947368	5	2.7	1	1	P
12	1.375	2.894736842	5	2.9	0	1.571428571	N
13	4.125	2.789473684	5	2.6	0	1.285714286	P
14	4.375	2.894736842	4.888888889	3	0	1.857142857	N
15	4.125	3.526315789	3.666666667	2.7	1	2.428571429	P

The above screenshot shows the predicted data using the KNN Classification.

2. (50 points) Python practice for KNN classification
Use the following data, and run KNN to find the best parameters and performance

- Use Malware detection data set (malware_Binary.csv) to run KNN by using 10-fold cross validation
- Read the description and ReadMe document in the malware detection data

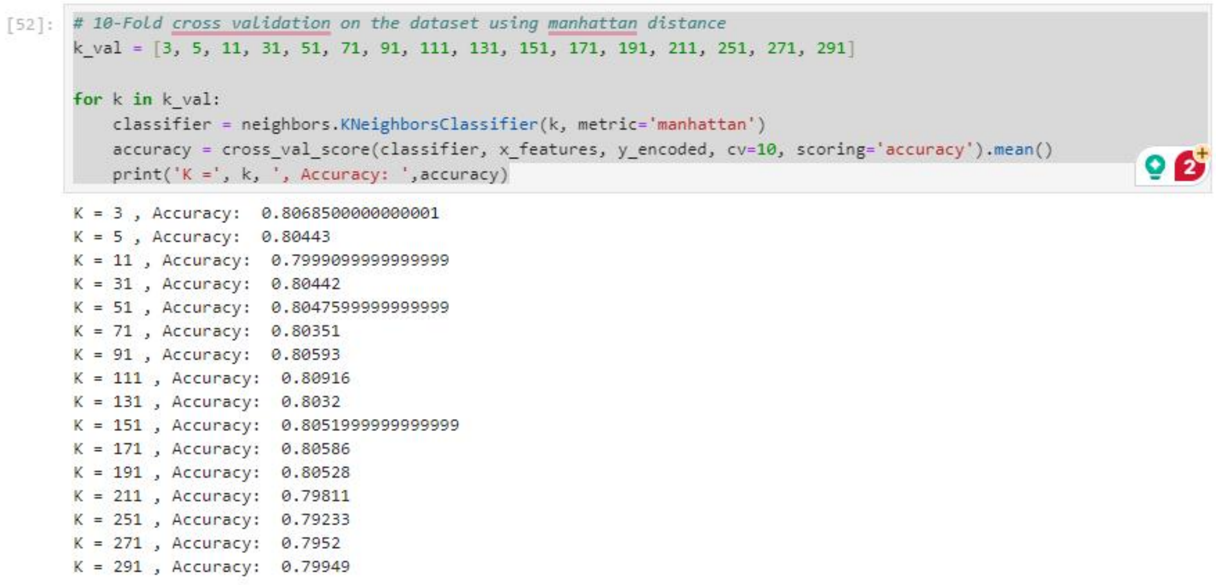
Note:

- You need to change different/multiple hyper-parameters to find the best KNN model.
- Use accuracy, precision, recall, AUC as metrics
- You can find data sets from "slide & data" on blackboard system

Submission

- The ipynb and saved html files
- A comparison of different parameters and AUC values in word or PDF file

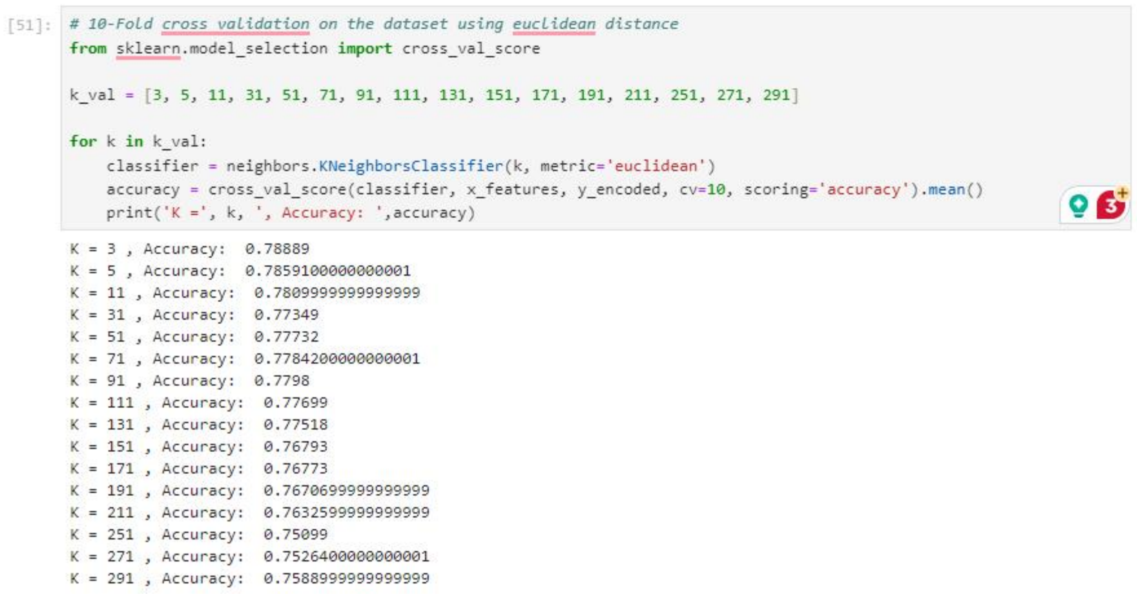
Ans-
In comparison K = 3 , Accuracy: 0.8068500000000001 using Manhattan distance



```
[52]: # 10-Fold cross validation on the dataset using manhattan distance
k_val = [3, 5, 11, 31, 51, 71, 91, 111, 131, 151, 171, 191, 211, 251, 271, 291]

for k in k_val:
    classifier = neighbors.KNeighborsClassifier(k, metric='manhattan')
    accuracy = cross_val_score(classifier, x_features, y_encoded, cv=10, scoring='accuracy').mean()
    print('K =', k, ', Accuracy: ',accuracy)
```

```
K = 3 , Accuracy:  0.8068500000000001
K = 5 , Accuracy:  0.80443
K = 11 , Accuracy:  0.7999099999999999
K = 31 , Accuracy:  0.80442
K = 51 , Accuracy:  0.8047599999999999
K = 71 , Accuracy:  0.80351
K = 91 , Accuracy:  0.80593
K = 111 , Accuracy:  0.80916
K = 131 , Accuracy:  0.8032
K = 151 , Accuracy:  0.8051999999999999
K = 171 , Accuracy:  0.80586
K = 191 , Accuracy:  0.80528
K = 211 , Accuracy:  0.79811
K = 251 , Accuracy:  0.79233
K = 271 , Accuracy:  0.7952
K = 291 , Accuracy:  0.79949
```

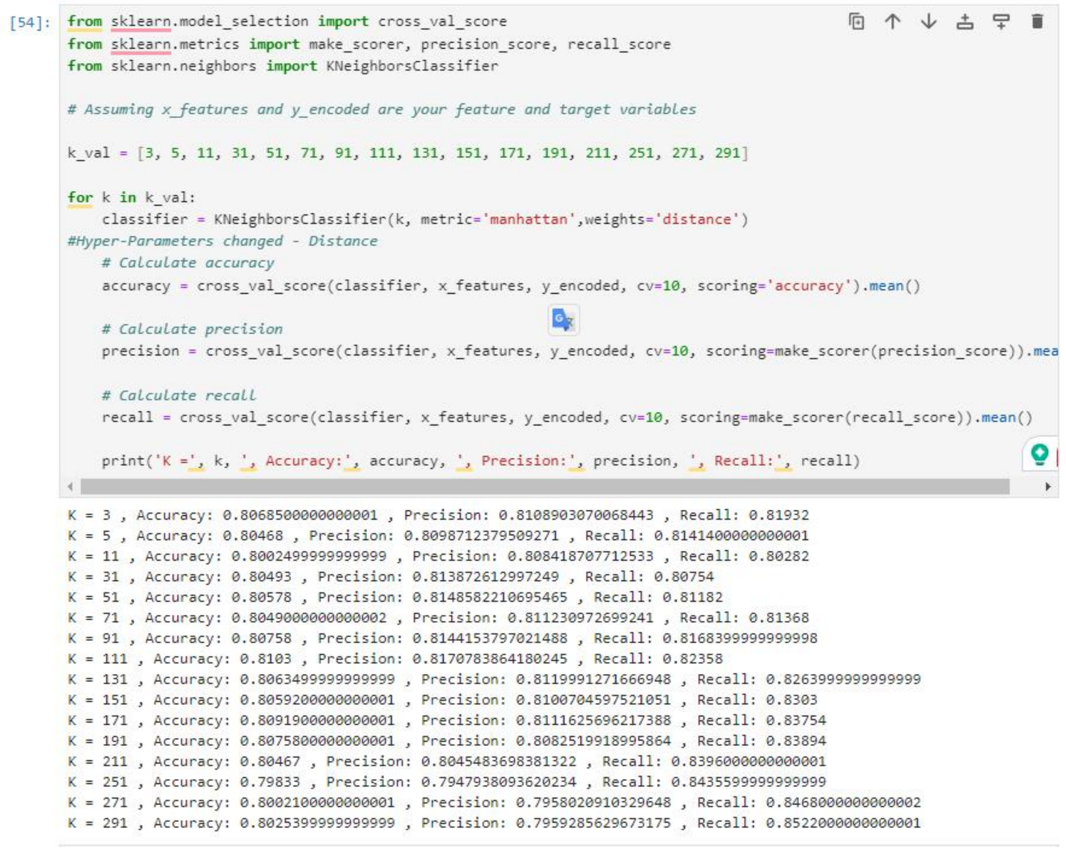
In comparison K = 3 , Accuracy: 0.78889 using Euclidean



```
[51]: # 10-Fold cross validation on the dataset using euclidean distance
      from sklearn.model_selection import cross_val_score

      k_val = [3, 5, 11, 31, 51, 71, 91, 111, 131, 151, 171, 191, 211, 251, 271, 291]

      for k in k_val:
          classifier = neighbors.KNeighborsClassifier(k, metric='euclidean')
          accuracy = cross_val_score(classifier, x_features, y_encoded, cv=10, scoring='accuracy').mean()
          print('K =', k, ', Accuracy: ',accuracy)
```

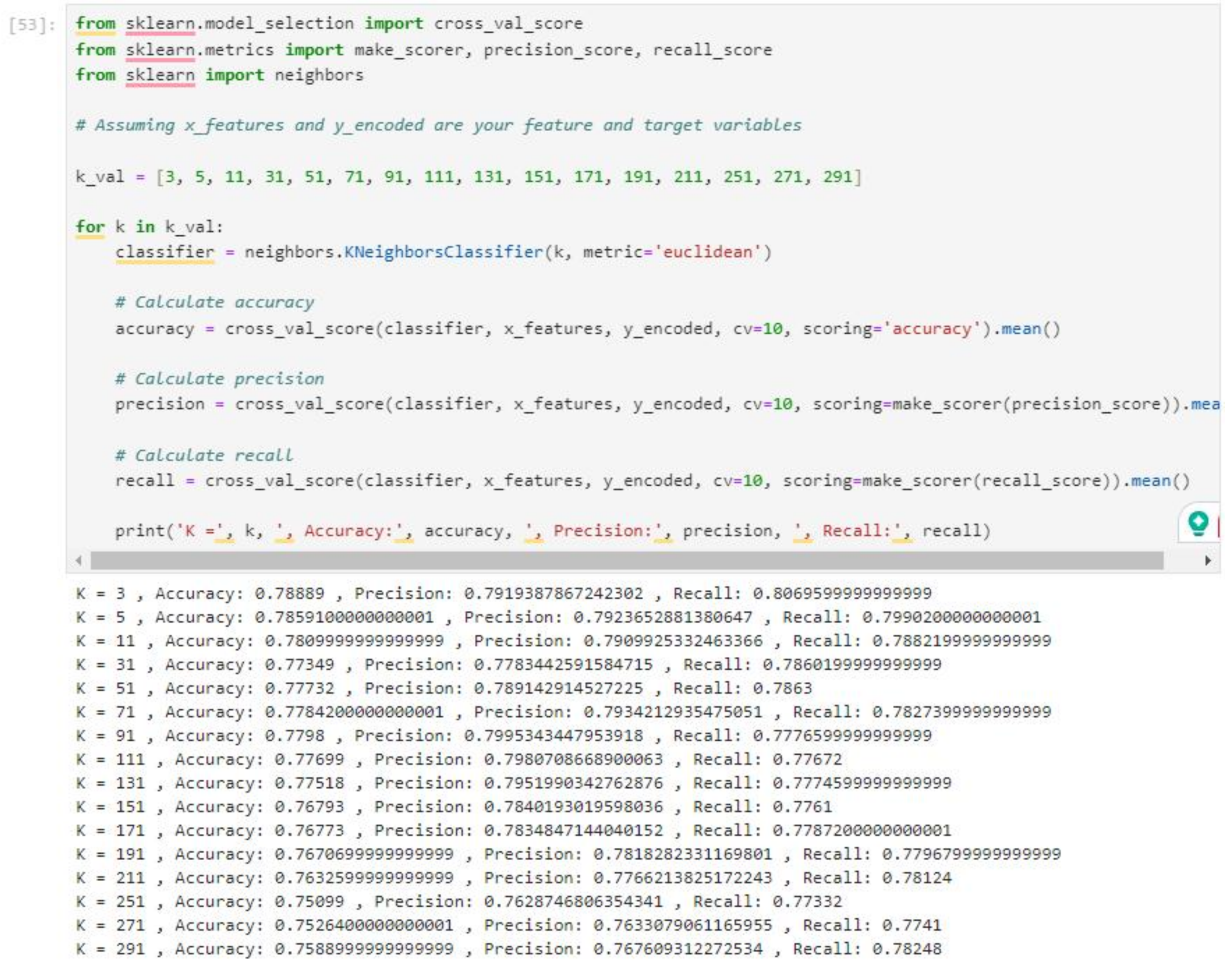
```
K = 3 , Accuracy:  0.78889
K = 5 , Accuracy:  0.7859100000000001
K = 11 , Accuracy:  0.7809999999999999
K = 31 , Accuracy:  0.77349
K = 51 , Accuracy:  0.77732
K = 71 , Accuracy:  0.7784200000000001
K = 91 , Accuracy:  0.7798
K = 111 , Accuracy:  0.77699
K = 131 , Accuracy:  0.77518
K = 151 , Accuracy:  0.76793
K = 171 , Accuracy:  0.76773
K = 191 , Accuracy:  0.7670699999999999
K = 211 , Accuracy:  0.7632599999999999
K = 251 , Accuracy:  0.75099
K = 271 , Accuracy:  0.7526400000000001
K = 291 , Accuracy:  0.7588999999999999
```

In the below screenshot you can see the change in hyper parameter-weights are added
K = 171 , Accuracy: 0.8091900000000001
K =5 ; Precision: 0.8098712379509271



```
[54]: from sklearn.model_selection import cross_val_score
      from sklearn.metrics import make_scorer, precision_score, recall_score
      from sklearn.neighbors import KNeighborsClassifier

      # Assuming x_features and y_encoded are your feature and target variables

      k_val = [3, 5, 11, 31, 51, 71, 91, 111, 131, 151, 171, 191, 211, 251, 271, 291]

      for k in k_val:
          classifier = KNeighborsClassifier(k, metric='manhattan',weights='distance')
      #Hyper-Parameters changed - Distance
          # Calculate accuracy
          accuracy = cross_val_score(classifier, x_features, y_encoded, cv=10, scoring='accuracy').mean()

          # Calculate precision
          precision = cross_val_score(classifier, x_features, y_encoded, cv=10, scoring=make_scorer(precision_score)).mea

          # Calculate recall
          recall = cross_val_score(classifier, x_features, y_encoded, cv=10, scoring=make_scorer(recall_score)).mean()

          print('K =', k, ', Accuracy:', accuracy, ', Precision:', precision, ', Recall:', recall)
```

```
K = 3 , Accuracy: 0.8068500000000001 , Precision: 0.8108903070068443 , Recall: 0.81932
K = 5 , Accuracy: 0.80468 , Precision: 0.8098712379509271 , Recall: 0.8141400000000001
K = 11 , Accuracy: 0.8002499999999999 , Precision: 0.808418707712533 , Recall: 0.80282
K = 31 , Accuracy: 0.80493 , Precision: 0.813872612997249 , Recall: 0.80754
K = 51 , Accuracy: 0.80578 , Precision: 0.8148582210695465 , Recall: 0.81182
K = 71 , Accuracy: 0.8049000000000002 , Precision: 0.811230972699241 , Recall: 0.81368
K = 91 , Accuracy: 0.80758 , Precision: 0.8144153797021488 , Recall: 0.8168399999999998
K = 111 , Accuracy: 0.8103 , Precision: 0.8170783864180245 , Recall: 0.82358
K = 131 , Accuracy: 0.8063499999999999 , Precision: 0.8119991271666948 , Recall: 0.8263999999999999
K = 151 , Accuracy: 0.8059200000000001 , Precision: 0.8100704597521051 , Recall: 0.8303
K = 171 , Accuracy: 0.8091900000000001 , Precision: 0.8111625696217388 , Recall: 0.83754
K = 191 , Accuracy: 0.8075800000000001 , Precision: 0.8082519918995864 , Recall: 0.83894
K = 211 , Accuracy: 0.80467 , Precision: 0.8045483698381322 , Recall: 0.8396000000000001
K = 251 , Accuracy: 0.79833 , Precision: 0.7947938093620234 , Recall: 0.8435599999999999
K = 271 , Accuracy: 0.8002100000000001 , Precision: 0.7958020910329648 , Recall: 0.8468000000000002
K = 291 , Accuracy: 0.8025399999999999 , Precision: 0.7959285629673175 , Recall: 0.8522000000000001
```



```
[53]: from sklearn.model_selection import cross_val_score
      from sklearn.metrics import make_scorer, precision_score, recall_score
      from sklearn import neighbors

      # Assuming x_features and y_encoded are your feature and target variables

      k_val = [3, 5, 11, 31, 51, 71, 91, 111, 131, 151, 171, 191, 211, 251, 271, 291]

      for k in k_val:
          classifier = neighbors.KNeighborsClassifier(k, metric='euclidean')

          # Calculate accuracy
          accuracy = cross_val_score(classifier, x_features, y_encoded, cv=10, scoring='accuracy').mean()

          # Calculate precision
          precision = cross_val_score(classifier, x_features, y_encoded, cv=10, scoring=make_scorer(precision_score)).mea

          # Calculate recall
          recall = cross_val_score(classifier, x_features, y_encoded, cv=10, scoring=make_scorer(recall_score)).mean()

          print('K =', k, ', Accuracy:', accuracy, ', Precision:', precision, ', Recall:', recall)
```

```
K = 3 , Accuracy: 0.78889 , Precision: 0.7919387867242302 , Recall: 0.8069599999999999
K = 5 , Accuracy: 0.7859100000000001 , Precision: 0.7923652881380647 , Recall: 0.7990200000000001
K = 11 , Accuracy: 0.7809999999999999 , Precision: 0.7909925332463366 , Recall: 0.7882199999999999
K = 31 , Accuracy: 0.77349 , Precision: 0.7783442591584715 , Recall: 0.7860199999999999
K = 51 , Accuracy: 0.77732 , Precision: 0.789142914527225 , Recall: 0.7863
K = 71 , Accuracy: 0.7784200000000001 , Precision: 0.7934212935475051 , Recall: 0.7827399999999999
K = 91 , Accuracy: 0.7798 , Precision: 0.7995343447953918 , Recall: 0.7776599999999999
K = 111 , Accuracy: 0.77699 , Precision: 0.7980708668900063 , Recall: 0.77672
K = 131 , Accuracy: 0.77518 , Precision: 0.7951990342762876 , Recall: 0.7774599999999999
K = 151 , Accuracy: 0.76793 , Precision: 0.7840193019598036 , Recall: 0.7761
K = 171 , Accuracy: 0.76773 , Precision: 0.7834847144040152 , Recall: 0.7787200000000001
K = 191 , Accuracy: 0.7670699999999999 , Precision: 0.7818282331169801 , Recall: 0.7796799999999999
K = 211 , Accuracy: 0.7632599999999999 , Precision: 0.7766213825172243 , Recall: 0.78124
K = 251 , Accuracy: 0.75099 , Precision: 0.7628746806354341 , Recall: 0.77332
K = 271 , Accuracy: 0.7526400000000001 , Precision: 0.7633079061165955 , Recall: 0.7741
K = 291 , Accuracy: 0.7588999999999999 , Precision: 0.767609312272534 , Recall: 0.78248
```

The above screenshot also explain with precision and Recall.

Precision is higher in K=91 where as Accuracy is higher in K=3