Final Report: KNN And Decision Tree

CSE-0408 Summer 2021

Rubyat Jesmin Shammi
Department of Computer Science and Engineering
State University of Bangladesh (SUB)
Dhaka, Bangladesh
rubyatshammiss@gmail.com

***Abstract*——This paper introduced for KNN problem and Decision Tree problem.**

***Index Terms*—**

I. Introduction for KNN

KNN is a lazy learning, non-parametric algorithm. It uses data with several classes to predict the classification of the new sample point. KNN is non-parametric since it doesn't make any assumptions on the data being studied, the model is distributed from the data.

II. Advantages for KNN

1.Simple to implement and intuitive to understand
2.Can learn non-linear decision boundaries when used for classfication and regression. Can came up with a highly flexible decision boundary adjusting the value of K.

III. Disadvantages for KNN

1.Performance Issue with large data-set: The time required to calculate the distance between the new point and each existing points is huge. Which then degrades the performance of the algorithm.
2.Value of K: It is really crucial to determine what value to assign to k. with different value of K you get different results

IV. Literature Review for knn

Author's introduction: Zhongheng Zhang, MMed. Department of Critical Care Medicine, Jinhua Municipal Central Hospital, Jinhua Hospital of Zhejiang University. Dr. Zhongheng Zhang is a fellow physician of the Jinhua Municipal Central Hospital. He graduated from School of Medicine, Zhejiang University in 2009, receiving Master Degree. He has published more than 35 academic papers (science citation indexed) that have been cited for over 200 times. He has been appointed as reviewer for 10 journals, including Journal of Cardiovascular Medicine, Hemodialysis International, Journal of Translational Medicine, Critical Care, International Journal of Clinical Practice, Journal of Critical Care.

V. Conclusion for KNN

KNN is an effective machine learning algorithm that can be used in credit scoring, prediction of cancer cells, image recognition, and many other applications. The main importance of using KNN is that it's easy to implement and works well with small data sets.

VI. Introduction for Decision tree

In a decision tree, the algorithm starts with a root node of a tree then compares the value of different attributes and follows the next branch until it reaches the end leaf node. It uses different algorithms to check about the split and variable that allow the best homogeneous sets of population.

VII. Advantages for Decision tree

1.When using Decision tree algorithm it is not necessary to normalize the data.
2.Decision tree algorithm implementation can be done without scaling the data as well.
3.When using Decision tree algorithm it is not necessary to impute the missing values.
4.The data pre-processing step for decision trees requires less code and analysis.
5.The data pre-processing step for decision trees requires less time.

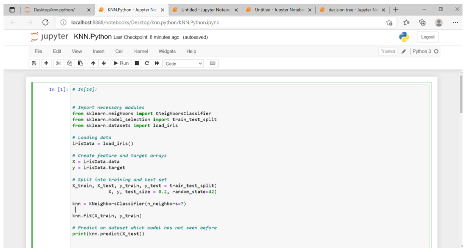
VIII. Disadvantages for Decision tree

1.The mathematical calculation of decision tree mostly require more memory.
2.The mathematical calculation of decision tree mostly require more time.
3.The space and time complexity of decision tree model is relatively higher.
4.Decision tree model training time is relatively more as complexity is high.

IX. Literature Review for decision tree

Angel Insua, Alberto Monje, Hom-Lay Wang, Marita Inglehart, Patient-Centered Perspectives and Understanding of Peri-Implantitis, Journal of Periodontology, 10.1902/jop.2017.160796, 88, 11, (1153-1162), (2017).

Wiley Online Library Nima D. Sarmast,Howard H. Wang,Nikolaos K. Soldatos,Nikola Angelov,Samuel Dorn,Raymond Yukna,Vincent J. Iacono, First published: 01 December 2016


Desktop/knn.python/
KNN.Python - Jupyter Not
Untitled - Jupyter Notebo
Untitled - Jupyter Notebo
decision tree - Jupyter No
localhost:8888/notebooks/Desktop/knn.python/KNN.Python.ipynb
jupyter
KNN.Python Last Checkpoint: 8 minutes ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code

```
In [1]: # In[14]:


        # Import necessary modules
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.model_selection import train_test_split
        from sklearn.datasets import load_iris

        # Loading data
        irisData = load_iris()

        # Create feature and target arrays
        X = irisData.data
        y = irisData.target

        # Split into training and test set
        X_train, X_test, y_train, y_test = train_test_split(
                     X, y, test_size = 0.2, random_state=42)

        knn = KNeighborsClassifier(n_neighbors=7)

        knn.fit(X_train, y_train)

        # Predict on dataset which model has not seen before
        print(knn.predict(X_test))
```

Fig. 1.

X. Conclusion for Decision tree

Decision trees assist analysts in evaluating upcoming choices. The tree creates a visual representation of all possible outcomes, rewards and follow-up decisions in one document. Each subsequent decision resulting from the original choice is also depicted on the tree, so you can see the overall effect of any one decision. As you go through the tree and make choices, you will see a specific path from one node to another and the impact a decision made now could have down the road.

Acknowledgment

I would like to thank my honourable **Khan Md. Hasib Sir** for his time, generosity and critical insights into this project.

References

[1] Zhou Q, Yang D, Ombrello AK, et al. Early-onset stroke and vasculopathy associated with mutations in ADA2. N Engl J Med. 2014;370:911–20.

[2] Navon Elkan P, Pierce SB, Segel R, et al. Mutant adenosine deaminase 2 in a polyarteritis nodosa vasculopathy. N Engl J Med. 2014;370:921–31.

[3] Oda H, Kastner DL. Genomics, biology, and human illness: advances in the monogenic autoinflammatory diseases. Rheum Dis Clin N Am. 2017;43:327–45.

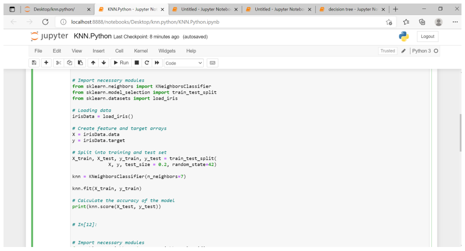
[4] Gonzalez Santiago TM, Zavialov A, Saarela J, et al. Dermatologic features of ADA2 deficiency in cutaneous polyarteritis nodosa. JAMA Dermatol. 2015;151:1230–4.

[5] Short RD, Fukunaga K. The optimal distance measure for nearest neighbor classification. IEEE Transactions on Information Theory 1981;27:622-7. 10.1109/TIT.1981.1056403 [CrossRef] [Google Scholar]

[6] Weinberger KQ, Saul LK. Distance metric learning for large margin nearest neighbor classification. The Journal of Machine Learning Research 2009;10:207-44. [Google Scholar]

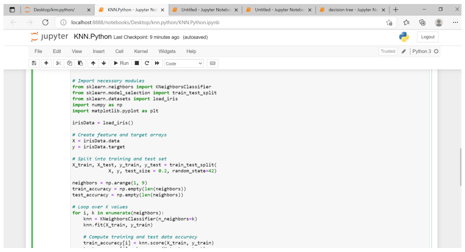
[7] Cost S, Salzberg S. A weighted nearest neighbor algorithm for learning with symbolic features. Machine Learning 1993;10:57-78. 10.1007/BF00993481 [CrossRef] [Google Scholar]

XI. Code


Desktop/knn.python/
KNN.Python - Jupyter Not
Untitled - Jupyter Notebo
Untitled - Jupyter Notebo
decision tree - Jupyter No
localhost:8888/notebooks/Desktop/knn.python/KNN.Python.ipynb
jupyter
KNN.Python Last Checkpoint: 8 minutes ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code

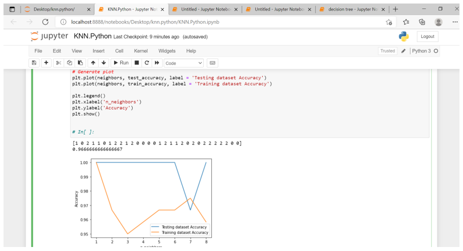
```
# Import necessary modules
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

# Loading data
irisData = load_iris()

# Create feature and target arrays
X = irisData.data
y = irisData.target

# Split into training and test set
X_train, X_test, y_train, y_test = train_test_split(
             X, y, test_size = 0.2, random_state=42)

knn = KNeighborsClassifier(n_neighbors=7)

knn.fit(X_train, y_train)

# Calculate the accuracy of the model
print(knn.score(X_test, y_test))


# In[12]:


# Import necessary modules
```

Fig. 2.


Desktop/knn.python/
KNN.Python - Jupyter Not
Untitled - Jupyter Notebo
Untitled - Jupyter Notebo
decision tree - Jupyter No
localhost:8888/notebooks/Desktop/knn.python/KNN.Python.ipynb
jupyter
KNN.Python Last Checkpoint: 9 minutes ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code

```
# Import necessary modules
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
import numpy as np
import matplotlib.pyplot as plt

irisData = load_iris()

# Create feature and target arrays
X = irisData.data
y = irisData.target

# Split into training and test set
X_train, X_test, y_train, y_test = train_test_split(
             X, y, test_size = 0.2, random_state=42)

neighbors = np.arange(1, 9)
train_accuracy = np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

# Loop over K values
for i, k in enumerate(neighbors):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)

    # Compute training and test data accuracy
    train_accuracy[i] = knn.score(X_train, y_train)
```

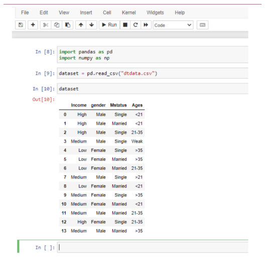
Fig. 3.


Desktop/knn.python/
KNN.Python - Jupyter Not
Untitled - Jupyter Notebo
Untitled - Jupyter Notebo
decision tree - Jupyter No
localhost:8888/notebooks/Desktop/knn.python/KNN.Python.ipynb
jupyter
KNN.Python Last Checkpoint: 9 minutes ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code
1.00
0.99
0.98
0.97
0.96
0.95
Accuracy
1
2
3
4
5
6
7
8
Testing dataset Accuracy
Training dataset Accuracy

```
# Generate plot
plt.plot(neighbors, test_accuracy, label = 'Testing dataset Accuracy')
plt.plot(neighbors, train_accuracy, label = 'Training dataset Accuracy')

plt.legend()
plt.xlabel('n_neighbors')
plt.ylabel('Accuracy')
plt.show()


# In[ ]:
```

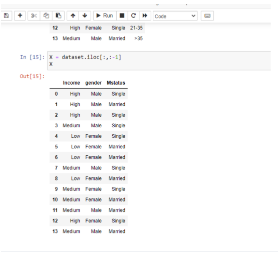
```
[1 0 2 1 1 0 1 2 2 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
0.9666666666666667
```

Fig. 4.



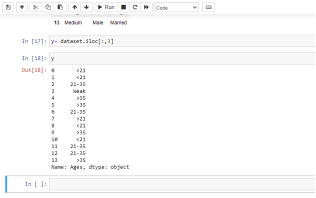
```
In [8]: import pandas as pd
        import numpy as np

In [9]: dataset = pd.read_csv("dtdata.csv")

In [10]: dataset

Out[10]:
     Income  gender  Mstatus   Ages
 0     High    Male   Single    <21
 1     High    Male  Married    <21
 2     High    Male   Single  21-35
 3   Medium    Male   Single   Weak
 4      Low  Female   Single    >35
 5      Low  Female  Married    >35
 6      Low  Female  Married  21-35
 7   Medium    Male   Single    >21
 8      Low  Female  Married    <21
 9   Medium  Female   Single    >35
10   Medium  Female  Married    <21
11   Medium    Male  Married  21-35
12     High  Female   Single  21-35
13   Medium    Male  Married    >35

In [ ]:
```

Fig. 5.



```
12     High  Female   Single  21-35
13   Medium    Male  Married    >35

In [15]: X = dataset.iloc[:,:-1]
         X

Out[15]:
     Income  gender  Mstatus
 0     High    Male   Single
 1     High    Male  Married
 2     High    Male   Single
 3   Medium    Male   Single
 4      Low  Female   Single
 5      Low  Female  Married
 6      Low  Female  Married
 7   Medium    Male   Single
 8      Low  Female  Married
 9   Medium  Female   Single
10   Medium  Female  Married
11   Medium    Male  Married
12     High  Female   Single
13   Medium    Male  Married
```

Fig. 6.



```
13  Medium    Male   Married

In [17]: y= dataset.iloc[:,3]

In [18]: y

Out[18]: 0       <21
         1       <21
         2     21-35
         3      Weak
         4       >35
         5       >35
         6     21-35
         7       >21
         8       <21
         9       >35
         10      <21
         11    21-35
         12    21-35
         13      >35
         Name: Ages, dtype: object

In [ ]:
```

Fig. 7.