Tucil1_13519111 _13519122

February 20, 2022

1 Decision Tree Learning

1.0.1 Member:

- Febriawan Ghally Ar Rahman (1359111)
- Mgs. Tabrani (13519122)

1.0.2 Content

1. DecisionTreeClassifier
2. Id3Estimator
3. K Means
4. LogisticRegression
5. Neural_network
6. SVM

1.1 1. Load Datasets

[319]:
```
import pandas as pd
from sklearn import datasets
import numpy as np
```

1.1.1 1.1 Load breast cancer dataset

[320]:
```
breast_cancer = datasets.load_breast_cancer()
X_breast_cancer, y_breast_cancer = datasets.load_breast_cancer(return_X_y=True)

# Display breast cancer dataframe
feature_names = list(breast_cancer['feature_names'])
feature_names.append('diagosis')
df_breast_cancer = pd.DataFrame(data= np.c_[breast_cancer['data'],
 ↪breast_cancer['target']], columns= feature_names)
df_breast_cancer
```

[320]:
```
     mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0          17.99         10.38          122.80     1001.0          0.11840
1          20.57         17.77          132.90     1326.0          0.08474
2          19.69         21.25          130.00     1203.0          0.10960
```

3	11.42	20.38	77.58	386.1	0.14250
4	20.29	14.34	135.10	1297.0	0.10030
..	…	…	…	…	…
564	21.56	22.39	142.00	1479.0	0.11100
565	20.13	28.25	131.20	1261.0	0.09780
566	16.60	28.08	108.30	858.1	0.08455
567	20.60	29.33	140.10	1265.0	0.11780
568	7.76	24.54	47.92	181.0	0.05263

	mean compactness	mean concavity	mean concave points	mean symmetry	\
0	0.27760	0.30010	0.14710	0.2419	
1	0.07864	0.08690	0.07017	0.1812	
2	0.15990	0.19740	0.12790	0.2069	
3	0.28390	0.24140	0.10520	0.2597	
4	0.13280	0.19800	0.10430	0.1809	
..	…	…	…	…	
564	0.11590	0.24390	0.13890	0.1726	
565	0.10340	0.14400	0.09791	0.1752	
566	0.10230	0.09251	0.05302	0.1590	
567	0.27700	0.35140	0.15200	0.2397	
568	0.04362	0.00000	0.00000	0.1587	

	mean fractal dimension	…	worst texture	worst perimeter	worst area	\
0	0.07871	…	17.33	184.60	2019.0	
1	0.05667	…	23.41	158.80	1956.0	
2	0.05999	…	25.53	152.50	1709.0	
3	0.09744	…	26.50	98.87	567.7	
4	0.05883	…	16.67	152.20	1575.0	
..	…	…	…	…	…	
564	0.05623	…	26.40	166.10	2027.0	
565	0.05533	…	38.25	155.00	1731.0	
566	0.05648	…	34.12	126.70	1124.0	
567	0.07016	…	39.42	184.60	1821.0	
568	0.05884	…	30.37	59.16	268.6	

	worst smoothness	worst compactness	worst concavity	\
0	0.16220	0.66560	0.7119	
1	0.12380	0.18660	0.2416	
2	0.14440	0.42450	0.4504	
3	0.20980	0.86630	0.6869	
4	0.13740	0.20500	0.4000	
..	…	…	…	
564	0.14100	0.21130	0.4107	
565	0.11660	0.19220	0.3215	
566	0.11390	0.30940	0.3403	
567	0.16500	0.86810	0.9387	
568	0.08996	0.06444	0.0000	

```
     worst concave points  worst symmetry  worst fractal dimension  diagosis
0                  0.2654          0.4601                  0.11890       0.0
1                  0.1860          0.2750                  0.08902       0.0
2                  0.2430          0.3613                  0.08758       0.0
3                  0.2575          0.6638                  0.17300       0.0
4                  0.1625          0.2364                  0.07678       0.0
..                    …               …                        …       …
564                0.2216          0.2060                  0.07115       0.0
565                0.1628          0.2572                  0.06637       0.0
566                0.1418          0.2218                  0.07820       0.0
567                0.2650          0.4087                  0.12400       0.0
568                0.0000          0.2871                  0.07039       1.0

[569 rows x 31 columns]
```

1.1.2 1.2 Load play tennis dataset

[321]:
```
df_play_tennis = pd.read_csv('data/play_tennis.csv')
df_play_tennis = df_play_tennis.drop(['day'],axis=1)

# Display play tennis dataframe
df_play_tennis
```

[321]:
```
     outlook  temp humidity    wind play
0      Sunny   Hot     High    Weak   No
1      Sunny   Hot     High  Strong   No
2   Overcast   Hot     High    Weak  Yes
3       Rain  Mild     High    Weak  Yes
4       Rain  Cool   Normal    Weak  Yes
5       Rain  Cool   Normal  Strong   No
6   Overcast  Cool   Normal  Strong  Yes
7      Sunny  Mild     High    Weak   No
8      Sunny  Cool   Normal    Weak  Yes
9       Rain  Mild   Normal    Weak  Yes
10     Sunny  Mild   Normal  Strong  Yes
11  Overcast  Mild     High  Strong  Yes
12  Overcast   Hot   Normal    Weak  Yes
13      Rain  Mild     High  Strong   No
```

1.2 2. Encode Categorical Data

[322]:
```
# Encode categorical data in play tennis dataframe
from sklearn import preprocessing
df_play_tennis = df_play_tennis.apply(preprocessing.LabelEncoder().
 ↪fit_transform)
```

```
# Divide play tennis dataframe to data and target
dataset_play_tennis = df_play_tennis.to_numpy()
X_play_tennis = []
y_play_tennis = []
for i in range(len(dataset_play_tennis)):
    X_play_tennis.append(dataset_play_tennis[i][:-1])
    y_play_tennis.append(dataset_play_tennis[i][-1])

# Display encoded dataframe
df_play_tennis
```

[322]:

	outlook	temp	humidity	wind	play
0	2	1	0	1	0
1	2	1	0	0	0
2	0	1	0	1	1
3	1	2	0	1	1
4	1	0	1	1	1
5	1	0	1	0	0
6	0	0	1	0	1
7	2	2	0	1	0
8	2	0	1	1	1
9	1	2	1	1	1
10	2	2	1	0	1
11	0	2	0	0	1
12	0	1	1	1	1
13	1	2	0	0	0

1.3 3. Split Datasets

[323]:

```
# Split dataset to 80% training data and 20% testing data
from sklearn.model_selection import train_test_split

# Split breast cancer dataset
X_training_breast_cancer, X_testing_breast_cancer =␣
↪train_test_split(X_breast_cancer, test_size=0.2, random_state=25)
y_training_breast_cancer, y_testing_breast_cancer =␣
↪train_test_split(y_breast_cancer, test_size=0.2, random_state=25)

# Split play tennis dataset
X_training_play_tennis, X_testing_play_tennis = train_test_split(X_play_tennis,␣
↪test_size=0.2, random_state=25)
y_training_play_tennis, y_testing_play_tennis = train_test_split(y_play_tennis,␣
↪test_size=0.2, random_state=25)
```

1.4 4. Learning with Decision Tree Classifier Algorithm

[324]:
```
from sklearn.tree import DecisionTreeClassifier
```

1.4.1 4.1 Breast Cancer

4.1.1 Metrics Evaluation

[325]:
```
clf_breast_cancer = DecisionTreeClassifier().fit(X_training_breast_cancer,␣
↪y_training_breast_cancer)
y_predict = clf_breast_cancer.predict(X_testing_breast_cancer)
print(metrics.classification_report(y_testing_breast_cancer, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.92      0.85      0.88        39
           1       0.92      0.96      0.94        75

    accuracy                           0.92       114
   macro avg       0.92      0.90      0.91       114
weighted avg       0.92      0.92      0.92       114
```

- f1-score for target ‘0’ (diagnosed as breast cancer) is 0.88
- f1-score for target ‘1’ (diagnosed as breast cancer) is 0.94
- accuracy for decision tree classifier model is 0.92 > We can conclude that another part of data given will probably predicted to be correct, because accuracy is high, then f1-scores are close to accuracy.

1.4.2 4.2 Play Tennis

4.2.1 Metrics Evaluation

[326]:
```
clf_play_tennis = DecisionTreeClassifier().fit(X_training_play_tennis,␣
↪y_training_play_tennis)
y_predict = clf_play_tennis.predict(X_testing_play_tennis)
print(metrics.classification_report(y_testing_play_tennis, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.50      1.00      0.67         1
           1       1.00      0.50      0.67         2

    accuracy                           0.67         3
   macro avg       0.75      0.75      0.67         3
weighted avg       0.83      0.67      0.67         3
```

- f1-score for target ‘0’ (decided as not play) is 0.67
- f1-score for target ‘1’ (decided as play) is 0.67

- accuracy for decision tree classifier model is 0.67 > We can conclude that this model is not as good as logistic regression and neural network models because it has lower accuracy and f1-scores, although accuracy is equal to f1-scores.

1.5 5. Learning with ID3 Estimator Algorithm

[327]:
```
import six
import sys
sys.modules['sklearn.externals.six'] = six
import mlrose
from id3 import Id3Estimator
from id3 import export_graphviz
estimator = Id3Estimator()
```

1.5.1 5.1 Breast Cancer

5.1.1 Decision Tree

[328]:
```
estimator = estimator.fit(X_training_breast_cancer, y_training_breast_cancer)
tree = export_graphviz(estimator.tree_, 'tree.dot', breast_cancer.feature_names)
```

5.1.2 Metrics Evaluation

[329]:
```
y_predict = estimator.predict(X_testing_breast_cancer)
print(metrics.classification_report(y_testing_breast_cancer, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.87      0.87      0.87        39
           1       0.93      0.93      0.93        75

    accuracy                           0.91       114
   macro avg       0.90      0.90      0.90       114
weighted avg       0.91      0.91      0.91       114
```

- f1-score for target ‘0’ (diagnosed as breast cancer) is 0.87
- f1-score for target ‘1’ (diagnosed as breast cancer) is 0.93
- accuracy for id3 estimator model is 0.91 > We can conclude that another part of data given will probably predicted to be correct, because accuracy is high, then f1-scores are close to accuracy.

1.5.2 5.2 Play tennis

5.2.1 Decision Tree

[330]:
```
estimator = estimator.fit(X_training_play_tennis, y_training_play_tennis)
tree = export_graphviz(estimator.tree_, 'tree.dot',␣
 ↪['outlook','temp','humidity','wind','play'])
```

5.2.2 Metrics Evaluation

[331]:
```
y_predict = estimator.predict(X_testing_play_tennis)
print(metrics.classification_report(y_testing_play_tennis, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.50      1.00      0.67         1
           1       1.00      0.50      0.67         2

    accuracy                           0.67         3
   macro avg       0.75      0.75      0.67         3
weighted avg       0.83      0.67      0.67         3
```

- f1-score for target '0' (decided as not play) is 0.67
- f1-score for target '1' (decided as play) is 0.67
- accuracy for id3 estimator model is 0.67 > We can conclude that this model is not as good as logistic regression and neural network models because it has lower accuracy and f1-scores, although accuracy is equal to f1-scores.

1.6 6. Learning with K Means Algorithm

[332]:
```
from sklearn.cluster import KMeans
from sklearn.metrics.cluster import homogeneity_score
```

1.6.1 6.1 Breast Cancer

6.1.1 Homogeneity

[333]:
```
kmeans = KMeans(n_clusters=2).fit(X_training_breast_cancer,
 ↪y_training_breast_cancer)
y_predict = kmeans.predict(X_testing_breast_cancer)
print(f"Homogeneity Score: {homogeneity_score(y_testing_breast_cancer,
 ↪y_predict)}")
```

```
Homogeneity Score: 0.4156347969069669
```

6.1.2 Metrics Evaluation

[334]:
```
print(metrics.classification_report(y_testing_breast_cancer, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.16      0.36      0.22        39
           1       0.04      0.01      0.02        75

    accuracy                           0.13       114
   macro avg       0.10      0.19      0.12       114
weighted avg       0.08      0.13      0.09       114
```

- f1-score for target '0' (diagnosed as breast cancer) is 0.22
- f1-score for target '1' (diagnosed as breast cancer) is 0.02
- accuracy for kmeans model is 0.13 > We can conclude that the accuracy of this model is not good rather than any models in this exploration.

1.6.2 6.2 Play Tennis

6.2.1 Homogeneity

[335]:
```
kmeans = KMeans(n_clusters=2).fit(X_training_play_tennis,␣
  ↪y_training_play_tennis)
y_predict = kmeans.predict(X_testing_play_tennis)
print(f"Homogeneity Score: {homogeneity_score(y_testing_play_tennis,␣
  ↪y_predict)}")
```

```
Homogeneity Score: 0.0
```

6.2.2 Metrics Evaluation

[336]:
```
import warnings
warnings.filterwarnings('ignore')
print(metrics.classification_report(y_testing_play_tennis, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.67      1.00      0.80         2

    accuracy                           0.67         3
   macro avg       0.33      0.50      0.40         3
weighted avg       0.44      0.67      0.53         3
```

- f1-score for target '0' (decided as not play) is 0.00
- f1-score for target '1' (decided as play) is 0.80
- accuracy for kmeans model is 0.67 > We can conclude that this model is not as good as logistic regression and neural network models because it has lower accuracy and f1-scores.

1.7 7. Learning with Logistic Regression Algorithm

[337]:
```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
```

1.7.1 7.1 Breast cancer

[338]:
```
clf_breast_cancer = LogisticRegression(random_state=0, max_iter=10000).
  ↪fit(X_training_breast_cancer, y_training_breast_cancer)
y_predict = clf_breast_cancer.predict(X_testing_breast_cancer)
```

7.1.1 Metrics Evaluation

[339]:
```python
print(metrics.classification_report(y_testing_breast_cancer, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.90      0.90      0.90        39
           1       0.95      0.95      0.95        75

    accuracy                           0.93       114
   macro avg       0.92      0.92      0.92       114
weighted avg       0.93      0.93      0.93       114
```

- f1-score for target '0' (diagnosed as breast cancer) is 0.90
- f1-score for target '1' (diagnosed as breast cancer) is 0.95
- accuracy for logistic regression model is 0.93 > We can conclude that another part of data given will probably predicted to be correct, because accuracy is high, then f1-scores are close to accuracy.

1.7.2 7.2 Play tennis

[340]:
```python
clf_play_tennis = LogisticRegression().fit(X_training_play_tennis,␣
 ↪y_training_play_tennis)
y_predict = clf_play_tennis.predict(X_testing_play_tennis)
```

7.2.1 Metrics Evaluation

[341]:
```python
print(metrics.classification_report(y_testing_play_tennis, y_predict))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         2

    accuracy                           1.00         3
   macro avg       1.00      1.00      1.00         3
weighted avg       1.00      1.00      1.00         3
```

- f1-score for target '0' (decided as not play) is 1.00
- f1-score for target '1' (decided as play) is 1.00
- accuracy for logistic regression model is 1.00 > We can conclude that another part of data given will almost predicted to be correct, because accuracy is equal to f1-scores.

1.8 8. Learning with Neural Network Algorithm

[342]:
```python
from sklearn.neural_network import MLPClassifier
```

1.8.1 8.1 Breast cancer

[343]:
```
clf_breast_cancer = MLPClassifier(random_state=1, max_iter=300).
 ↪fit(X_training_breast_cancer, y_training_breast_cancer)
y_predict = clf_breast_cancer.predict(X_testing_breast_cancer)
```

8.1.1 Metrics Evaluation

[344]:
```
print(metrics.classification_report(y_testing_breast_cancer, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.97      0.90      0.93        39
           1       0.95      0.99      0.97        75

    accuracy                           0.96       114
   macro avg       0.96      0.94      0.95       114
weighted avg       0.96      0.96      0.96       114
```

- f1-score for target '0' (diagnosed as breast cancer) is 0.93
- f1-score for target '1' (diagnosed as breast cancer) is 0.97
- accuracy for neural network model is 0.96 > We can conclude that another part of data given will probably predicted to be correct, because accuracy is high, then f1-scores are close to accuracy.

1.8.2 8.2 Play tennis

[345]:
```
clf_play_tennis = MLPClassifier(random_state=1, max_iter=1000).
 ↪fit(X_training_play_tennis, y_training_play_tennis)
y_predict = clf_play_tennis.predict(X_testing_play_tennis)
```

8.2.1 Metrics Evaluation

[346]:
```
print(metrics.classification_report(y_testing_play_tennis, y_predict))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         2

    accuracy                           1.00         3
   macro avg       1.00      1.00      1.00         3
weighted avg       1.00      1.00      1.00         3
```

- f1-score for target '0' (decided as not play) is 1.00
- f1-score for target '1' (decided as play) is 1.00
- accuracy for neural network model is 1.00 > We can conclude that another part of data given will almost predicted to be correct, because accuracy is equal to f1-scores.

1.9 9. Learning with SVM algorithm

[347]:
```
from sklearn.svm import SVC
```

1.9.1 9.1 Breast cancer

[348]:
```
clf_play_tennis = SVC().fit(X_training_breast_cancer, y_training_breast_cancer)
y_predict = clf_play_tennis.predict(X_testing_breast_cancer)
```

9.1.1 Metrics Evaluation

[349]:
```
print(metrics.classification_report(y_testing_breast_cancer, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.97      0.77      0.86        39
           1       0.89      0.99      0.94        75

    accuracy                           0.91       114
   macro avg       0.93      0.88      0.90       114
weighted avg       0.92      0.91      0.91       114
```

- f1-score for target '0' (diagnosed as breast cancer) is 0.86
- f1-score for target '1' (diagnosed as breast cancer) is 0.94
- accuracy for SVM model is 0.91 > We can conclude that another part of data given will probably predicted to be correct, because accuracy is high, then f1-scores are close to accuracy.

1.9.2 9.2 Play tennis

[350]:
```
clf_play_tennis = SVC().fit(X_training_play_tennis, y_training_play_tennis)
y_predict = clf_play_tennis.predict(X_testing_play_tennis)
```

9.2.1 Metrics Evaluation

[351]:
```
print(metrics.classification_report(y_testing_play_tennis, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.50      1.00      0.67         1
           1       1.00      0.50      0.67         2

    accuracy                           0.67         3
   macro avg       0.75      0.75      0.67         3
weighted avg       0.83      0.67      0.67         3
```

- f1-score for target '0' (decided as not play) is 0.67
- f1-score for target '1' (decided as play) is 0.67

- accuracy for SVM model is 0.67 > We can conclude that this model is not as good as logistic regression and neural network models because it has lower accuracy and f1-scores, although accuracy is equal to f1-scores.