workshop-03-tasks

March 14, 2021

1 Demo

- Data pre-processing
- Nearest neighbors based classification

[30]:

```
# Import the commonly-used modules
import pandas as pd
import numpy as np

# Load the dataset
data = pd.read_csv("/home/dalia/Master of Information Technology in␣
 ↪Cybersecurity/Session 3/COMP8325 Artificial Intelligence/week_3/Week 3␣
 ↪Workshop-20210310/data/titanic.csv")
print("data size: "+str(data.shape))
data.head()
```

```
data size: (891, 12)
```

[30]:

```
   PassengerId                                               Name     Sex  \
0            1                            Braund, Mr. Owen Harris    male
1            2  Cumings, Mrs. John Bradley (Florence Briggs Th…  female
2            3                             Heikkinen, Miss. Laina  female
3            4       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female
4            5                           Allen, Mr. William Henry    male

    Age  SibSp  Parch            Ticket     Fare Cabin Embarked  Pclass  \
0  22.0      1      0         A/5 21171   7.2500   NaN        S       3
1  38.0      1      0          PC 17599  71.2833   C85        C       1
2  26.0      0      0  STON/O2. 3101282   7.9250   NaN        S       3
3  35.0      1      0            113803  53.1000  C123        S       1
4  35.0      0      0            373450   8.0500   NaN        S       3

   Survived
0         0
1         1
2         1
3         1
```

```
4         0
```

[31]:
```
# size of the data
print("(row, column):",data.shape)
```

```
(row, column): (891, 12)
```

[32]:
```
# check for missing values
data.isnull().sum()
```

[32]:
```
PassengerId      0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
Pclass           0
Survived         0
dtype: int64
```

[33]:
```
# one option is to just simply drop rows with any missing values
data_no_missing_value=data.dropna()
print("(row, column) after dropping missing values:\n",data_no_missing_value.
 ↪shape)
```

```
(row, column) after dropping missing values:
 (183, 12)
```

[34]:
```
# get one column from the data
data_name=data_no_missing_value["Name"]
data_name
```

[34]:
```
1      Cumings, Mrs. John Bradley (Florence Briggs Th…
3           Futrelle, Mrs. Jacques Heath (Lily May Peel)
6                                McCarthy, Mr. Timothy J
10                       Sandstrom, Miss. Marguerite Rut
11                              Bonnell, Miss. Elizabeth
                             …
871     Beckwith, Mrs. Richard Leonard (Sallie Monypeny)
872                             Carlsson, Mr. Frans Olof
879        Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)
887                         Graham, Miss. Margaret Edith
889                                Behr, Mr. Karl Howell
```

```
Name: Name, Length: 183, dtype: object
```

[35]:
```
# get three columns from the data
data_sex_age_survived=data_no_missing_value[['Sex', 'Age', 'Survived']]
data_sex_age_survived
```

[35]:
```
        Sex   Age  Survived
1    female  38.0         1
3    female  35.0         1
6      male  54.0         0
10   female   4.0         1
11   female  58.0         1
..      …     …         …
871  female  47.0         1
872    male  33.0         0
879  female  56.0         1
887  female  19.0         1
889    male  26.0         1

[183 rows x 3 columns]
```

[36]:
```
# encode categorical attributes
data_sex_age_survived['Sex']=data_sex_age_survived.Sex.astype('category').cat.
 ↪codes
data_sex_age_survived
```

```
<ipython-input-36-9aac392383e8>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_sex_age_survived['Sex']=data_sex_age_survived.Sex.astype('category').cat.
codes
```

[36]:
```
     Sex   Age  Survived
1      0  38.0         1
3      0  35.0         1
6      1  54.0         0
10     0   4.0         1
11     0  58.0         1
..   …     …         …
871    0  47.0         1
872    1  33.0         0
879    0  56.0         1
887    0  19.0         1
889    1  26.0         1
```

```
[183 rows x 3 columns]
```

[37]:

```
# define X and y
feature_columns=['Sex', 'Age']
X=data_sex_age_survived[feature_columns]
y=data_sex_age_survived.Survived
print(X.head())
print(y.head())
```

```
    Sex   Age
1     0  38.0
3     0  35.0
6     1  54.0
10    0   4.0
11    0  58.0
1     1
3     1
6     0
10    1
11    1
Name: Survived, dtype: int64
```

[38]:

```
# train/test data split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

[39]:

```
# train a NN classifier
from sklearn import neighbors
clf = neighbors.KNeighborsClassifier(n_neighbors=1)
clf.fit(X_train, y_train)
```

[39]:

```
KNeighborsClassifier(n_neighbors=1)
```

[40]:

```
# predict the test data
y_pred=clf.predict(X_test)
y_pred
```

[40]:

```
array([0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
       0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1])
```

[41]:

```
# calculate testing accuracy
from sklearn import metrics
print(metrics.accuracy_score(y_test, y_pred))
```

```
0.8043478260869565
```

```
[ ]:
```

```
[ ]:
```

2 Task 1: Build a k-NN classifier

```
[74]: # Load the dataset
data = pd.read_csv("/home/dalia/Master of Information Technology in␣
 ↪Cybersecurity/Session 3/COMP8325 Artificial Intelligence/week_3/Week 3␣
 ↪Workshop-20210310/data/titanic.csv")
print("data size: "+str(data.shape))
data.head()
```

```
data size: (891, 12)
```

```
[74]:    PassengerId                                               Name     Sex  \
0            1                            Braund, Mr. Owen Harris    male
1            2  Cumings, Mrs. John Bradley (Florence Briggs Th…  female
2            3                             Heikkinen, Miss. Laina  female
3            4       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female
4            5                           Allen, Mr. William Henry    male

    Age  SibSp  Parch            Ticket     Fare Cabin Embarked  Pclass  \
0  22.0      1      0         A/5 21171   7.2500   NaN        S       3
1  38.0      1      0          PC 17599  71.2833   C85        C       1
2  26.0      0      0  STON/O2. 3101282   7.9250   NaN        S       3
3  35.0      1      0            113803  53.1000  C123        S       1
4  35.0      0      0            373450   8.0500   NaN        S       3

   Survived
0         0
1         1
2         1
3         1
4         0
```

```
[75]: # delete the attribute "Cabin"
del data['Cabin']
```

```
[76]: # fill missing values with the mean age
print("The mean age: ", data.Age.mean())

data.Age.fillna(data.Age.mean(), inplace=True)

#check for mmissing values
data.isnull().sum()
```

```
The mean age:  29.69911764705882
```

[76]:

```
PassengerId    0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       2
Pclass         0
Survived       0
dtype: int64
```

[77]:

```python
# drop rows with any missing values (the "Embarked" atrribute)

data = data[data.Embarked.notnull()]

#check missing values
print(data.isnull().sum())

print("Data size with no missing values: "+str(data.shape))
```

```
PassengerId    0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
Pclass         0
Survived       0
dtype: int64
Data size with no missing values: (889, 11)
```

A question: Are “PassengerId”, “Name” and “Ticket” suitable for being a feature?

[78]:

```python
# Select all other attributes to form a data set
data_reduced = pd.DataFrame(data.drop(['PassengerId', 'Name', 'Ticket'],␣
 ↪axis='columns'))
data_reduced.head()
```

[78]:

```
      Sex   Age  SibSp  Parch     Fare Embarked  Pclass  Survived
0    male  22.0      1      0   7.2500        S       3         0
1  female  38.0      1      0  71.2833        C       1         1
```

```
2  female  26.0      0      0   7.9250        S       3         1
3  female  35.0      1      0  53.1000        S       1         1
4    male  35.0      0      0   8.0500        S       3         0
```

[79]:
```
# encode the categorical attributes where applicable
data_reduced['Sex'] = data_reduced['Sex'].astype('category').cat.codes
data_reduced['Embarked'] = data_reduced['Embarked'].astype('category').cat.codes
data_reduced.head()
```

[79]:
```
   Sex   Age  SibSp  Parch     Fare  Embarked  Pclass  Survived
0    1  22.0      1      0   7.2500         2       3         0
1    0  38.0      1      0  71.2833         0       1         1
2    0  26.0      0      0   7.9250         2       3         1
3    0  35.0      1      0  53.1000         2       1         1
4    1  35.0      0      0   8.0500         2       3         0
```

[80]:
```
# build a 1-NN classification model where "Survived" is the prediction target

X = data_reduced.drop('Survived', axis='columns')
y = data_reduced['Survived']

# train and test data split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y)

# build a 1-NN classification model

clf = neighbors.KNeighborsClassifier(n_neighbors = 1)
clf.fit(X_train, y_train)

y_pred=clf.predict(X_test)

print(metrics.accuracy_score(y_test, y_pred))
```

```
0.6995515695067265
```

[ ]:
```

```

3 Task 2 Tune the hyper parameters

[93]:
```
# mannually try other k values, e.g., 2, 3, 5, 10, the accuracy comparison

# build a 2-NN classification model
clf = neighbors.KNeighborsClassifier(n_neighbors = 2)
clf.fit(X_train, y_train)

```

```
y_pred=clf.predict(X_test)

print(metrics.accuracy_score(y_test, y_pred))
```

```
0.7130044843049327
```

[94]:

```
# build a 3-NN classification model
clf = neighbors.KNeighborsClassifier(n_neighbors = 3)
clf.fit(X_train, y_train)

y_pred=clf.predict(X_test)

print(metrics.accuracy_score(y_test, y_pred))
```

```
0.7533632286995515
```

[95]:

```
# build a 5-NN classification model
clf = neighbors.KNeighborsClassifier(n_neighbors = 5)
clf.fit(X_train, y_train)

y_pred=clf.predict(X_test)

print(metrics.accuracy_score(y_test, y_pred))
```

```
0.7354260089686099
```

[96]:

```
# build a 10-NN classification model
clf = neighbors.KNeighborsClassifier(n_neighbors = 10)
clf.fit(X_train, y_train)

y_pred=clf.predict(X_test)

print(metrics.accuracy_score(y_test, y_pred))
print(y_test)
print(y_pred)
```

```
0.7174887892376681
141    1
169    0
95     0
726    1
242    0
      ..
609    1
206    0
668    0
461    0
400    1
```

```
Name: Survived, Length: 223, dtype: int64
[0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 1 1 1 0
 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 1 0 1 0 0 0 0
 0 0 1 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0
 0 0 1 0 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0
 1 1 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0
 0]
```

[98]:

```python
# use k-fold cross-validation (10-fold cross validation)
from sklearn.model_selection import cross_val_score

n_values = range(1, 11)
for n in n_values:
    clf = neighbors.KNeighborsClassifier(n_neighbors = n)
    scores = cross_val_score(clf, X, y, cv=10, scoring='accuracy')
    print("Accuracy: %0.2f.(+/- %0.2f)" %(scores.mean(),scores.std()))
```

```
Accuracy: 0.69.(+/- 0.04)
Accuracy: 0.70.(+/- 0.04)
Accuracy: 0.72.(+/- 0.05)
Accuracy: 0.70.(+/- 0.04)
Accuracy: 0.70.(+/- 0.03)
Accuracy: 0.69.(+/- 0.03)
Accuracy: 0.72.(+/- 0.05)
Accuracy: 0.69.(+/- 0.04)
Accuracy: 0.71.(+/- 0.05)
Accuracy: 0.70.(+/- 0.04)
```

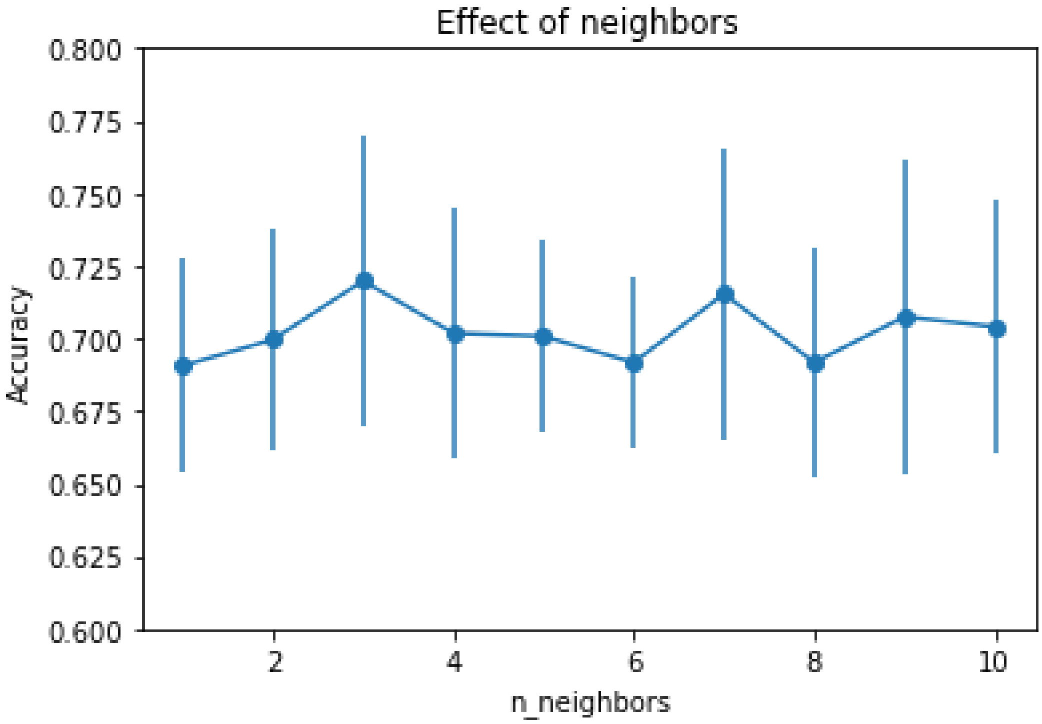
[86]:

```python
# visualize
import matplotlib.pyplot as plt

accuracy_scores = list()
accuracy_scores_std = list()

for n in n_values:
    clf = neighbors.KNeighborsClassifier(n_neighbors = n)
    scores = cross_val_score(clf, X, y, cv=10, scoring='accuracy')
    accuracy_scores.append(scores.mean())
    accuracy_scores_std.append(scores.std())

plt.errorbar(n_values, accuracy_scores, yerr=accuracy_scores_std, marker='o')
plt.title('Effect of neighbors')
plt.xlabel('n_neighbors')
plt.ylabel('Accuracy')
plt.ylim(0.6, 0.8)
plt.show()
```



[99]:

```
# Use grid search to faciliate the identification of the best hyper parameter
from sklearn.model_selection import GridSearchCV

tuned_parameters = [{'n_neighbors': range(1,10)}]

search = GridSearchCV(neighbors.KNeighborsClassifier(), tuned_parameters,␣
 ↪scoring='accuracy', cv=10)
search.fit(X_train, y_train)

print("Best parameters set found on training set:")
print(search.best_params_)
```

```
Best parameters set found on training set:
{'n_neighbors': 8}
```

[ ]:

4 Task 3 Build a decision tree model for the above classification task

[ ]:

[ ]:

```
```

[ ]:

```
```

[ ]:

```
```