COP528 Applied Machine Learning (LABs)

LAB Day 01. Fundamentals

Introduction

The aim of lab session is for you to understand the basic operations of several libraries for machine learning, including data reading and writing, simple classification and regression calculations, data visualisation, and the calculation of evaluation metrics. It includes **4** tutorial examples and **3** tasks. You may follow the tutorials to learn the basic function operations and then complete the tasks.

Tutorial 01. Pandas

Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The DataFrame is one of these structures. Follow the tutorial to learn functions that you could use to load data and look for some insights of the data.

Step 1. Import the necessary libraries

```
import pandas as pd
```

Step 2. Import the dataset by using Pandas function.

```
#pd.read_csv() reads a csv file into pandas DataFrame objects
users=pd.read_csv('https://raw.githubusercontent.com/justmarkham/DAT8/master/data/u.user',
sep='|', index_col='user_id')
```

Step 3. Print the first 10 entries.

```
users.head(10)
```

Step 4. Print the last 10 entries.

```
users.tail(10)
```

Step 5. What is the number of observations in the dataset?

```
users.shape[0]
```

Step 6. What is the number of attributes in the dataset?

```
users.shape[1]
```

Step 7. Print the name of all the attributes.

```
users.columns
```

Step 8. What is the data type of each attributes?

```
users.dtypes
```

Step 9. Print only the xxxx attribute column

```
#An example about the gender attribute
users.gender
```

Step 10. How many different occupations are in this dataset?

```
users.occupation.nunique()
```

Step 11. What is the most frequent occupation?

```
users.occupation.value_counts().head(1).index[0]
```

Step 12. Summarize the DataFrame.

```
users.describe()
```

Step 13. Summarize all the columns.

```
users.describe(include = "all")
```

Step 14. Summarize only the occupation column.

```
users.occupation.describe()
```

Step 15. What is the mean age of users?

```
round(users.age.mean())
```

Tutorial 02. From Pandas to Numpy Array

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. In one of the most popular machine learning package sklearn, many classifiers and clustering algorithms use numpy array as their input.

Follow the tutorial to learn some basic functions for array manipulation in Numpy.

Step 1. Create a random numerical DataFrame

```
import numpy as np
import pandas as pd
df = pd.DataFrame(np.random.randn(10, 4), columns=["A", "B",
"C", "D"])
df
```

Step 2. DataFrame is compatible with basic NumPy functions

```
np.exp(df) #elementwise exponential
np.sqrt(df) #elementwise square root
```

Step 3. DataFrame to NumPy arrays

```
df_all = np.asarray(df) #convert all the DF rows into array
df_all.shape
df_part = np.asarray(df[4:7]) #convert only 3 rows
df_part.shape
```

Step 4. Apply a simple sklearn function on NumPy array (here, we use a linear regression which is one of the most common regression models)

```
from sklearn.linear_model import LinearRegression
X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
# y = 1 * x_0 + 2 * x_1 + 3
y = np.dot(X, np.array([1, 2])) + 3

# train a linear regressor
reg = LinearRegression().fit(X, y)

# display coefficient of the prediction
reg.score(X, y)

# predict a new sample
reg.predict(np.array([[3, 5]]))
```

Step 5. Save and load model

```
import pickle

#save our regressor in the current directory
pkl_filename = "regressor.pkl"
with open(pkl_filename, 'wb') as file:
    pickle.dump(reg, file)

#load from file
with open(pkl_filename, 'rb') as file:
    reg_checkpoint = pickle.load(file)

#should get same result as before
reg_checkpoint.predict(np.array([[3, 5]]))
```

Tutorial 03. Simple visualization of data (some fundamental of matplotlib)

Matplotlib is one of the most popular data visualisation packages for Python and runs cross-platform. It is a common 2D plotting library for Python, but it also provides a partial 3D plotting interface. Matplotlib is often used in conjunction with NumPy and Pandas and is one of the most important tools for data analysis.

Follow the tutorial to learn some simple functions you can use to visualise the data.

Step 1. Create a 2D-plot with matplitlib

```
import matplotlib.pyplot as plt
import numpy as np
X = np.array([1,2,3,4])
Y = np.array([2,1,5,3])

plt.plot(X, Y)
plt.show()
```

Step 2. Create a grid of subplots

```
fig = plt.figure()  # an empty figure with no Axes
fig, ax = plt.subplots()  # a figure with a single Axes
plt.show()
fig, axs = plt.subplots(2, 2)  # a figure with a 2x2 grid of Axes
axs[0,0].plot(X,Y) # plot y=f(x) on first ax
plt.show()
```

Step 3. Editing figure and axis

```
fig, axs = plt.subplots(2, 2)   # a figure with a 2x2 grid of Axes
axs[0,0].plot(X,Y, label='xy_curve')
axs[0,0].set_xlabel('x label')   # set x axis label for subplot 1
axs[0,0].set_ylabel('y label')
axs[0,0].legend() # display legends
axs[0,0].set_title('XY_plot')   # set title for subplot 1
plt.show()
```

Task 01. preliminary analysis of datasets

Python for machine learning: sklearn is the most popular machine learning library used in python. It has many functions, including clustering methods, and classification methods as well as evaluation metrics and toy datasets for you to practice. For more detailed information about sklearn see https://scikit-learn.org/stable/

The California housing dataset is a classic dataset in sklearn.datasets. Please provide a preliminary analysis based on your learning today.

Here is the code to load the data and put them into a Panda dataframe.

```
from sklearn.datasets import fetch_california_housing

california_housing = fetch_california_housing(as_frame=True)
```

Tutorial 04. Performance evaluation

Performance evaluation is for finding out how good your model performs on a task and comparing the results from different models. Sklearn provides you with many methods that could be called to evaluate the results.

Follow the tutorial to make sure you understand how to use the functions for your model evaluation.

(1) Calculate the **accuracy** of your model:

```
import numpy as np
from sklearn.metrics import accuracy_score
y_true = [0, 1, 2, 3]
y_pred = [0, 2, 1, 3]
print(accuracy_score(y_true, y_pred))
```

(2) The **confusion_matrix** function evaluates classification accuracy by computing the confusion matrix with each row corresponding to the true.

```
from sklearn.metrics import confusion_matrix
y_true = [2, 0, 2, 2, 0, 1]
y_pred = [0, 0, 2, 2, 0, 2]
confusion_matrix(y_true, y_pred)
```

(3) For binary problems, we can get counts of **true negatives**, **false positives**, **false negatives** and **true positives** as follows:

```
y_true = [0, 0, 0, 1, 1, 1, 1, 1]
y_pred = [0, 1, 0, 1, 0, 1, 0, 1]
tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
tn, fp, fn, tp
```

(4) For **calculating precision**, **recall** and **F1**, you could use the following functions:

```
from sklearn import metrics
y_true = [0, 1, 0, 1]
y_pred = [0, 1, 0, 0]
metrics.precision_score(y_true, y_pred)
metrics.recall_score(y_true, y_pred)
metrics.f1_score(y_true, y_pred)
```

(5) The function **roc_curve** computes the **receiver operating characteristic curve** or **ROC curve.**

```
import numpy as np
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

y = np.array([1, 1, 2, 2])
scores = np.array([0.1, 0.4, 0.35, 0.8])
fpr, tpr, thresholds = roc_curve(y, scores, pos_label=2)
print(fpr)
print(tpr)
print(thresholds)
auc_score = roc_auc_score(y, scores)
print(auc_score)

from matplotlib import pyplot as plt
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr, label='ROC')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()
```

(6) For calculating **Root Mean Squared Error (RMSE)**, **Mean Absolute Error (MAE)** and **R-Squared**, you could use the following functions:

```
import numpy as np
from sklearn import metrics
y_true = [0, 1, 2, 4, 8]
y_pred = [0, 1, 3, 5, 7]

RMSE=np.sqrt(metrics.mean_squared_error(y_true,y_pred))
MAE=metrics.mean_absolute_error(y_true, y_pred)
R2=metrics.r2_score(y_true, y_pred)
#r2_score is not a symmetric function. Note the order of input.
```

Task 02. Breast Cancer dataset

The breast Cancer dataset is a binary classification task. The following code is using a simple logistic regression classifier to predict the class of each sample. Please use the metrics you learn, e.g., accuracy, precision, recall, F1, AUC and ROC_curve, to evaluate the performance of the classifier.

```
from sklearn.datasets import load_breast_cancer
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import numpy as np

X, y = load_breast_cancer(return_X_y=True)

X_train,X_test,y_train,y_test=train_test_split(
      X,y,test_size=0.2, random_state=0)

lr=LogisticRegression(solver='lbfgs', max_iter=8000)
lr.fit(X_train,y_train)
y_pred=lr.predict(X_test)
y_conf=lr.predict_proba(X_test)
```

Task 03. California_housing dataset

California_housing is a classic dataset for the house price prediction problem. The code below regresses the Median_house_value for each sample using a simple linear regression model. please use metrics you have learned, e.g. Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and R-Squared to evaluate the performance of the model.

```
from sklearn.linear_model import LinearRegression
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
#import dataset
california=fetch_california_housing()
x=california["data"]
y=california["target"]
names=california["feature_names"]
#Dividing the data set and training set
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=125)
# Build a linear regression model and train a test set
reg_linear = LinearRegression().fit (x_train, y_train)
y_pred_linear=reg_linear.predict(x_test)
```