ML with Python (part1)

Installing Anaconda

Link to download Anaconda

https://www.anaconda.com/distribution/#windows

Steps to download Anaconda:

https://docs.anaconda.com/anaconda/install/windows/

Installing PyCharm Edu

https://www.jetbrains.com/pycharm-edu/

Exploring Anaconda and PyCharm

To be illustrated in the lecture.

Common Libraries

NumPy: It has advanced math functions, scientific computing package and useful features for operations on n-arrays and matrices in Python.

SciPy: SciPy is a library of software for engineering and science. SciPy contains modules for linear algebra, optimization, integration, and statistics. The main functionality of SciPy library is built upon NumPy, and thus, its arrays make substantial use of NumPy.

Pandas: It designed for quick and easy data manipulation, aggregation, and visualization.

Matplotlib: For visualization

Seaborn: For advanced visualization

SciKit-Learn:
Designed for specific functionalities like image processing and machine learning facilitation.

Theano, ***PyTorch,*** **TensorFlow**, and **Keras**: are used for Deep Learning**.**

NLTK and **Gensim**: are used for Natural Language Processing.

Selecting dataset

Iris.csv

Some basic operations over the dataset

Load The Data

We are using pandas to load the data. We will also use pandas next to explore the data both with descriptive statistics and data visualization.

```
import pandas

dataset = pandas.read_csv("C:\\Users\\USER\\iris_dataset.csv")
```

Dimensions of Dataset

```
print(dataset.shape)
```

Top rows of the Data

```
print(dataset.head(5))
```

Statistical Summary

```
print(dataset.describe( ))
```

ML with Python (part2)

Create a Validation Dataset

```
from sklearn import model_selection
array = dataset.values
X = array[:,0:4]
Y = array[:,4]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation =
model_selection.train_test_split(X, Y, test_size=validation_size,
random_state=seed)
```

Make Predictions

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, Y_train)
predictions = knn.predict(X_validation)
from sklearn.metrics import accuracy_score
print(accuracy_score(Y_validation, predictions))
```

ML with Python part2 (Extension)

Make Predictions

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
from sklearn.metrics import accuracy_score
print(accuracy_score(Y_validation, predictions))



from sklearn import svm
model = svm.SVC()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
from sklearn.metrics import accuracy_score
print(accuracy_score(Y_validation, predictions))



from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
from sklearn.metrics import accuracy_score
print(accuracy_score(Y_validation, predictions))
```

ML with Python (part3)

K-fold Cross Validation

Using the train/test method of estimating the skill of the procedure on unseen data sometimes has a high variance. This means that when it is repeated, it gives different results, often very different results.

Cross-validation is another method to estimate the skill of a method on unseen data. Like using a train-test split. Cross-validation is a statistical method used to estimate the skill of machine learning models. It is commonly used in applied machine learning to compare and select a model for a given predictive modelling problem because it is easy to understand, and easy to implement.

This, in turn, provides a population of performance measures.

- We can calculate the mean of these measures to get an idea of how well the procedure performs on average.
- We can calculate the standard deviation of these measures to get an idea of how much the skill of the procedure is expected to vary in practice.

The train-test split and k-fold cross validation are called resampling methods. Resampling methods are statistical procedures for sampling a dataset and estimating an unknown quantity.

Your model will likely perform better when trained on all of the available data than just the subset used to estimate the performance of the model.

The general procedure is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
   - Take the group as a hold out or test data set
   - Take the remaining groups as a training data set
   - Fit a model on the training set and evaluate it on the test set
   - Retain the evaluation score and discard the model
4. Summarize the skill of the model using the sample of model evaluation scores

ML with Python (part4)

Python code for the K-fold Cross Validation

```
dataset = pandas.read_csv("C:\\iris_dataset.csv")

array = dataset.values
X = array[:,0:4]
Y = array[:,4]
from sklearn import model_selection
kfold = model_selection.KFold(n_splits=10)

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
cv_results = model_selection.cross_val_score(knn, X, Y, cv=kfold)
msg = "Mean = %f STD= %f" % (cv_results.mean(), cv_results.std())
print(msg)


from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
cv_results = model_selection.cross_val_score(dtc, X, Y, cv=kfold)
msg = "Mean = %f STD= %f" % (cv_results.mean(), cv_results.std())
print(msg)
```