



Classification with Python

In this notebook we try to practice all the classification algorithms that we have learned in this course.

We load a dataset using Pandas library, and apply the following algorithms, and find the best one for this specific dataset by accuracy evaluation methods.

Let's first load required libraries:

```
In [ ]: import itertools
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import NullFormatter
import pandas as pd
import numpy as np
import matplotlib.ticker as ticker
from sklearn import preprocessing
%matplotlib inline
```

About dataset

This dataset is about past loans. The **Loan_train.csv** data set includes details of 346 customers whose loan are already paid off or defaulted. It includes following fields:

Field	Description
Loan_status	Whether a loan is paid off on in collection
Principal	Basic principal loan amount at the
Terms	Origination terms which can be weekly (7 days), biweekly, and monthly payoff schedule
Effective_date	When the loan got originated and took effects
Due_date	Since it's one-time payoff schedule, each loan has one single due date
Age	Age of applicant
Education	Education of applicant
Gender	The gender of applicant

Let's download the dataset

```
In [ ]: !wget -O loan_train.csv https://cf-courses-data.s3.us.cloud-object-storage.appdomain
```

Load Data From CSV File

```
In [ ]: df = pd.read_csv('loan_train.csv')
```

```
df.head()
```

```
In [ ]: df.shape
```

Convert to date time object

```
In [ ]: df['due_date'] = pd.to_datetime(df['due_date'])
df['effective_date'] = pd.to_datetime(df['effective_date'])
df.head()
```

Data visualization and pre-processing

Let's see how many of each class is in our data set

```
In [ ]: df['loan_status'].value_counts()
```

260 people have paid off the loan on time while 86 have gone into collection

Let's plot some columns to understand data better:

```
In [ ]: # notice: installing seaborn might takes a few minutes
!conda install -c anaconda seaborn -y
```

```
In [ ]: import seaborn as sns

bins = np.linspace(df.Principal.min(), df.Principal.max(), 10)
g = sns.FacetGrid(df, col="Gender", hue="loan_status", palette="Set1", col_wrap=2)
g.map(plt.hist, 'Principal', bins=bins, ec="k")

g.axes[-1].legend()
plt.show()
```

```
In [ ]: bins = np.linspace(df.age.min(), df.age.max(), 10)
g = sns.FacetGrid(df, col="Gender", hue="loan_status", palette="Set1", col_wrap=2)
g.map(plt.hist, 'age', bins=bins, ec="k")

g.axes[-1].legend()
plt.show()
```

Pre-processing: Feature selection/extraction

Let's look at the day of the week people get the loan

```
In [ ]: df['dayofweek'] = df['effective_date'].dt.dayofweek
bins = np.linspace(df.dayofweek.min(), df.dayofweek.max(), 10)
g = sns.FacetGrid(df, col="Gender", hue="loan_status", palette="Set1", col_wrap=2)
g.map(plt.hist, 'dayofweek', bins=bins, ec="k")
g.axes[-1].legend()
plt.show()
```

We see that people who get the loan at the end of the week don't pay it off, so let's use

Feature binarization to set a threshold value less than day 4

```
In [ ]: df['weekend'] = df['dayofweek'].apply(lambda x: 1 if (x>3) else 0)
df.head()
```

Convert Categorical features to numerical values

Let's look at gender:

```
In [ ]: df.groupby(['Gender'])['loan_status'].value_counts(normalize=True)
```

86 % of female pay there loans while only 73 % of males pay there loan

Let's convert male to 0 and female to 1:

```
In [ ]: df['Gender'].replace(to_replace=['male', 'female'], value=[0,1], inplace=True)
df.head()
```

One Hot Encoding

How about education?

```
In [ ]: df.groupby(['education'])['loan_status'].value_counts(normalize=True)
```

Features before One Hot Encoding

```
In [ ]: df[['Principal', 'terms', 'age', 'Gender', 'education']].head()
```

Use one hot encoding technique to conver categorical variables to binary variables and append them to the feature Data Frame

```
In [ ]: Feature = df[['Principal', 'terms', 'age', 'Gender', 'weekend']]
Feature = pd.concat([Feature, pd.get_dummies(df['education'])], axis=1)
Feature.drop(['Master or Above'], axis = 1, inplace=True)
Feature.head()
```

Feature Selection

Let's define feature sets, X:

```
In [ ]: X = Feature
X[0:5]
```

What are our lables?

```
In [ ]: y = df['loan_status'].values
y[0:5]
```

Normalize Data

Data Standardization give data zero mean and unit variance (technically should be done

after train test split)

```
In [ ]: X= preprocessing.StandardScaler().fit(X).transform(X)  
X[0:5]
```

Classification

Now, it is your turn, use the training set to build an accurate model. Then use the test set to report the accuracy of the model You should use the following algorithm:

- K Nearest Neighbor(KNN)
- Decision Tree
- Support Vector Machine
- Logistic Regression

__ Notice: __

- You can go above and change the pre-processing, feature selection, feature-extraction, and so on, to make a better model.
- You should use either scikit-learn, Scipy or Numpy libraries for developing the classification algorithms.
- You should include the code of the algorithm in the following cells.

K Nearest Neighbor(KNN)

Notice: You should find the best k to build the model with the best accuracy.\ **warning:** You should not use the **loan_test.csv** for finding the best k, however, you can split your train_loan.csv into train and test to find the best **k**.

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

Decision Tree

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

Support Vector Machine

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

Logistic Regression

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

Model Evaluation using Test set

```
In [ ]: from sklearn.metrics import jaccard_score
from sklearn.metrics import f1_score
from sklearn.metrics import log_loss
```

First, download and load the test set:

```
In [ ]: !wget -O loan_test.csv https://s3-api.us-gio.objectstorage.softlayer.net/cf-courses
```

Load Test set for evaluation

```
In [ ]: test_df = pd.read_csv('loan_test.csv')
test_df.head()
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

Report

You should be able to report the accuracy of the built model using different evaluation metrics:

Algorithm	Jaccard	F1-score	LogLoss
KNN	?	?	NA
Decision Tree	?	?	NA
SVM	?	?	NA
LogisticRegression	?	?	?

Want to learn more?

IBM SPSS Modeler is a comprehensive analytics platform that has many machine learning algorithms. It has been designed to bring predictive intelligence to decisions made by individuals, by groups, by systems – by your enterprise as a whole. A free trial is available through this course, available here: [SPSS Modeler](#)

Also, you can use Watson Studio to run these notebooks faster with bigger datasets. Watson Studio is IBM's leading cloud solution for data scientists, built by data scientists. With Jupyter notebooks, RStudio, Apache Spark and popular libraries pre-packaged in the cloud, Watson Studio enables data scientists to collaborate on their projects without having to install anything. Join the fast-growing community of Watson Studio users today with a free account at [Watson Studio](#)

Thanks for completing this lesson!

Author: [Saeed Aghabozorgi](#)

[Saeed Aghabozorgi](#), PhD is a Data Scientist in IBM with a track record of developing enterprise level applications that substantially increases clients' ability to turn data into actionable knowledge. He is a researcher in data mining field and expert in developing advanced analytic methods like machine learning and statistical modelling on large datasets.

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-10-27	2.1	Lakshmi Holla	Made changes in import statement due to updates in version of sklearn library
2020-08-27	2.0	Malika Singla	Added lab to GitLab

© IBM Corporation 2020. All rights reserved.