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 [1]:
```
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 [2]:
```
!wget -O loan_train.csv https://cf-courses-da
```

```
--2022-06-06 20:06:20--  https://cf-courses-da
ta.s3.us.cloud-object-storage.appdomain.cloud/
IBMDeveloperSkillsNetwork-ML0101EN-SkillsNetwo
rk/labs/FinalModule_Coursera/data/loan_train.c
sv
Resolving cf-courses-data.s3.us.cloud-object-s
torage.appdomain.cloud (cf-courses-data.s3.us.
cloud-object-storage.appdomain.cloud)... 169.6
3.118.104
Connecting to cf-courses-data.s3.us.cloud-obje
ct-storage.appdomain.cloud (cf-courses-data.s
3.us.cloud-object-storage.appdomain.cloud)|16
9.63.118.104|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 23101 (23K) [text/csv]
Saving to: ‘loan_train.csv’

loan_train.csv      100%[===================>]
22.56K  --.-KB/s    in 0s

2022-06-06 20:06:20 (284 MB/s) - ‘loan_train.c
sv’ saved [23101/23101]
```

Load Data From CSV File

In [3]:
```
df = pd.read_csv('loan_train.csv')
df.head()
```

Out[3]:

	Unnamed: 0	Unnamed: 0.1	loan_status	Principal	terms
0	0	0	PAIDOFF	1000	30
1	2	2	PAIDOFF	1000	30
2	3	3	PAIDOFF	1000	15
3	4	4	PAIDOFF	1000	30
4	6	6	PAIDOFF	1000	30

In [4]:
```
df.shape
```

Out[4]:
```
(346, 10)
```

Convert to date time object

In [5]:
```
df['due_date'] = pd.to_datetime(df['due_date'
df['effective_date'] = pd.to_datetime(df['eff
df.head()
```

Out[5]:

	Unnamed: 0	Unnamed: 0.1	loan_status	Principal	terms
0	0	0	PAIDOFF	1000	30
1	2	2	PAIDOFF	1000	30
2	3	3	PAIDOFF	1000	15
3	4	4	PAIDOFF	1000	30
4	6	6	PAIDOFF	1000	30

Data visualization and pre-processing

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

In [6]:
```
df['loan_status'].value_counts()
```

Out[6]:
```
PAIDOFF       260
COLLECTION     86
Name: loan_status, dtype: int64
```

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

Let's plot some columns to underestand data better:

In [7]:
```
# notice: installing seaborn might takes a fe
%pip install seaborn
%pip install scikit-learn==0.23.1
```

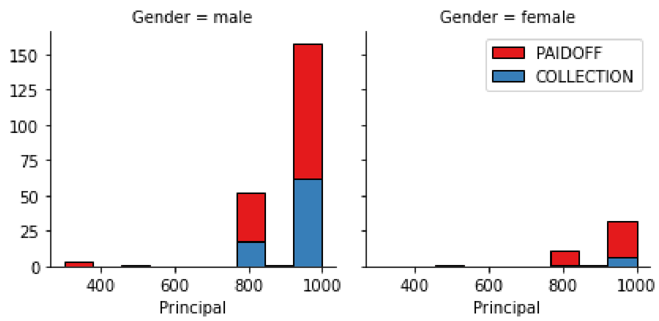
```
Requirement already satisfied: seaborn in /hom
e/jupyterlab/conda/envs/python/lib/python3.7/s
ite-packages (0.9.0)
Requirement already satisfied: scipy>=0.14.0 i
n /home/jupyterlab/conda/envs/python/lib/pytho
n3.7/site-packages (from seaborn) (1.7.3)
Requirement already satisfied: pandas>=0.15.2
in /home/jupyterlab/conda/envs/python/lib/pyth
on3.7/site-packages (from seaborn) (1.3.5)
Requirement already satisfied: matplotlib>=1.
4.3 in /home/jupyterlab/conda/envs/python/lib/
python3.7/site-packages (from seaborn) (3.5.2)
```

```
Requirement already satisfied: numpy>=1.9.3 in
/home/jupyterlab/conda/envs/python/lib/python
3.7/site-packages (from seaborn) (1.21.6)
Requirement already satisfied: python-dateutil
>=2.7 in /home/jupyterlab/conda/envs/python/li
b/python3.7/site-packages (from matplotlib>=1.
4.3->seaborn) (2.8.2)
Requirement already satisfied: packaging>=20.0
in /home/jupyterlab/conda/envs/python/lib/pyth
on3.7/site-packages (from matplotlib>=1.4.3->s
eaborn) (21.3)
Requirement already satisfied: cycler>=0.10 in
/home/jupyterlab/conda/envs/python/lib/python
3.7/site-packages (from matplotlib>=1.4.3->sea
born) (0.11.0)
Requirement already satisfied: pyparsing>=2.2.
1 in /home/jupyterlab/conda/envs/python/lib/py
thon3.7/site-packages (from matplotlib>=1.4.3-
>seaborn) (3.0.9)
Requirement already satisfied: pillow>=6.2.0 i
n /home/jupyterlab/conda/envs/python/lib/pytho
n3.7/site-packages (from matplotlib>=1.4.3->se
aborn) (8.1.0)
Requirement already satisfied: kiwisolver>=1.
0.1 in /home/jupyterlab/conda/envs/python/lib/
python3.7/site-packages (from matplotlib>=1.4.
3->seaborn) (1.4.2)
Requirement already satisfied: fonttools>=4.2
2.0 in /home/jupyterlab/conda/envs/python/lib/
python3.7/site-packages (from matplotlib>=1.4.
3->seaborn) (4.33.3)
Requirement already satisfied: pytz>=2017.3 in
/home/jupyterlab/conda/envs/python/lib/python
3.7/site-packages (from pandas>=0.15.2->seabor
n) (2022.1)
Requirement already satisfied: typing-extensio
ns in /home/jupyterlab/conda/envs/python/lib/p
ython3.7/site-packages (from kiwisolver>=1.0.1
->matplotlib>=1.4.3->seaborn) (4.2.0)
Requirement already satisfied: six>=1.5 in /ho
me/jupyterlab/conda/envs/python/lib/python3.7/
site-packages (from python-dateutil>=2.7->matp
lotlib>=1.4.3->seaborn) (1.16.0)
Note: you may need to restart the kernel to us
e updated packages.
Requirement already satisfied: scikit-learn==
0.23.1 in /home/jupyterlab/conda/envs/python/l
ib/python3.7/site-packages (0.23.1)
Requirement already satisfied: scipy>=0.19.1 i
n /home/jupyterlab/conda/envs/python/lib/pytho
n3.7/site-packages (from scikit-learn==0.23.1)
(1.7.3)
Requirement already satisfied: numpy>=1.13.3 i
n /home/jupyterlab/conda/envs/python/lib/pytho
n3.7/site-packages (from scikit-learn==0.23.1)
(1.21.6)
Requirement already satisfied: joblib>=0.11 in
/home/jupyterlab/conda/envs/python/lib/python
3.7/site-packages (from scikit-learn==0.23.1)
(1.1.0)
Requirement already satisfied: threadpoolctl>=
2.0.0 in /home/jupyterlab/conda/envs/python/li
b/python3.7/site-packages (from scikit-learn==
0.23.1) (3.1.0)
Note: you may need to restart the kernel to us
```

e updated packages.

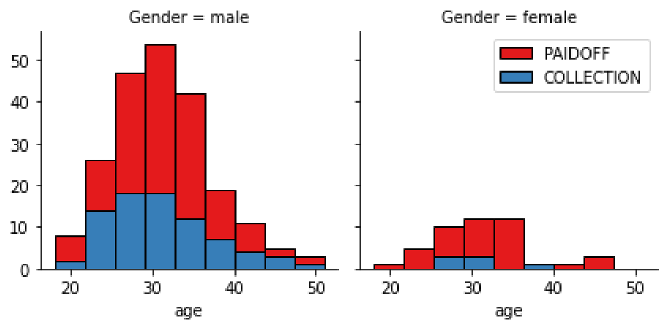
In [8]:

```
import seaborn as sns

bins = np.linspace(df.Principal.min(), df.Pri
g = sns.FacetGrid(df, col="Gender", hue="loan
g.map(plt.hist, 'Principal', bins=bins, ec="k

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



In [9]:

```
bins = np.linspace(df.age.min(), df.age.max()
g = sns.FacetGrid(df, col="Gender", hue="loan
g.map(plt.hist, 'age', bins=bins, ec="k")

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

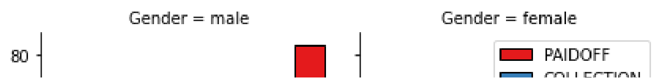


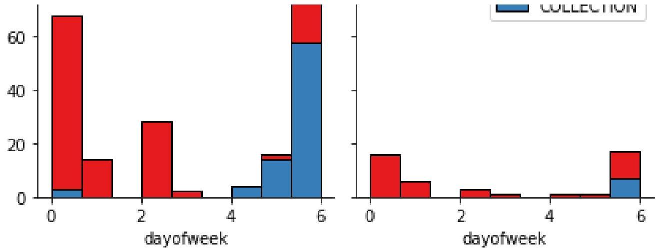
Pre-processing: Feature selection/extraction

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

In [10]:

```
df['dayofweek'] = df['effective_date'].dt.day
bins = np.linspace(df.dayofweek.min(), df.day
g = sns.FacetGrid(df, col="Gender", hue="loan
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 [11]:

```
df['weekend'] = df['dayofweek'].apply(lambda
df.head()
```

Out[11]:

	Unnamed: 0	Unnamed: 0.1	loan_status	Principal	terms
0	0	0	PAIDOFF	1000	30
1	2	2	PAIDOFF	1000	30
2	3	3	PAIDOFF	1000	15
3	4	4	PAIDOFF	1000	30
4	6	6	PAIDOFF	1000	30

Convert Categorical features to numerical values

Let's look at gender:

In [12]:

```
df.groupby(['Gender'])['loan_status'].value_c
```

Out[12]:

```
Gender  loan_status
female  PAIDOFF        0.865385
        COLLECTION     0.134615
male    PAIDOFF        0.731293
        COLLECTION     0.268707
Name: loan_status, dtype: float64
```

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 [13]:

```
df['Gender'].replace(to_replace=['male','fema
```

```
df.head()
```

Out[13]:

	Unnamed: 0	Unnamed: 0.1	loan_status	Principal	terms
0	0	0	PAIDOFF	1000	30
1	2	2	PAIDOFF	1000	30
2	3	3	PAIDOFF	1000	15
3	4	4	PAIDOFF	1000	30
4	6	6	PAIDOFF	1000	30

One Hot Encoding

How about education?

In [14]:
```
df.groupby(['education'])['loan_status'].valu
```

Out[14]:
```
education             loan_status
Bechalor              PAIDOFF        0.750000
                      COLLECTION     0.250000
High School or Below  PAIDOFF        0.741722
                      COLLECTION     0.258278
Master or Above       COLLECTION     0.500000
                      PAIDOFF        0.500000
college               PAIDOFF        0.765101
                      COLLECTION     0.234899
Name: loan_status, dtype: float64
```

Features before One Hot Encoding

In [15]:
```
df[['Principal','terms','age','Gender','educa
```

Out[15]:

	Principal	terms	age	Gender	education
0	1000	30	45	0	High School or Below
1	1000	30	33	1	Bechalor
2	1000	15	27	0	college
3	1000	30	28	1	college
4	1000	30	29	0	college

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

In [16]:

```
Feature = df[['Principal','terms','age','Gend
Feature = pd.concat([Feature,pd.get_dummies(d
Feature.drop(['Master or Above'], axis = 1,in
Feature.columns
```

Out[16]:

```
Index(['Principal', 'terms', 'age', 'Gender',
'weekend', 'Bechalor',
       'High School or Below', 'college'],
      dtype='object')
```

Feature Selection

Let's define feature sets, X:

In [17]:

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

Out[17]:

	Principal	terms	age	Gender	weekend	Bechalor
0	1000	30	45	0	0	0
1	1000	30	33	1	0	1
2	1000	15	27	0	0	0
3	1000	30	28	1	1	0
4	1000	30	29	0	1	0

What are our lables?

In [18]:

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

Out[18]:

```
array(['PAIDOFF', 'PAIDOFF', 'PAIDOFF', 'PAIDO
FF', 'PAIDOFF'],
      dtype=object)
```

Normalize Data

Data Standardization give data zero mean and unit variance (technically should be done after train test split)

In [19]:

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

Out[19]:

```
array([[ 0.51578458,  0.92071769,  2.33152555,
-0.42056004, -1.20577805,
        -0.38170062,  1.13639374, -0.8696810
8],
```

```
       [ 0.51578458,  0.92071769,  0.34170148,
2.37778177, -1.20577805,
         2.61985426, -0.87997669, -0.8696810
8],
       [ 0.51578458, -0.95911111, -0.65321055,
-0.42056004, -1.20577805,
        -0.38170062, -0.87997669,  1.1498467
9],
       [ 0.51578458,  0.92071769, -0.48739188,
2.37778177,  0.82934003,
        -0.38170062, -0.87997669,  1.1498467
9],
       [ 0.51578458,  0.92071769, -0.3215732 ,
-0.42056004,  0.82934003,
        -0.38170062, -0.87997669,  1.1498467
9]])
```

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.

Importing all the packages

In [20]:

```
from sklearn.model_selection import train_tes
from sklearn.metrics import f1_score
from sklearn.metrics import jaccard_score
from sklearn.neighbors import KNeighborsClass
from sklearn import metrics
from sklearn.metrics import classification_re
import itertools
from sklearn.linear_model import LogisticRegr
```

```
from sklearn.metrics import log_loss
from sklearn.tree import DecisionTreeClassifi
import sklearn.tree as tree
from sklearn import svm
```

K Nearest Neighbor(KNN)

All data passed to a dataframe at the end of file

In [21]:

```
#split up the data
X_train, X_test, y_train, y_test = train_test
print ('Train set:', X_train.shape,  y_train.
print ('Test set:', X_test.shape,  y_test.sha
```

```
Train set: (276, 8) (276,)
Test set: (70, 8) (70,)
```

In [22]:

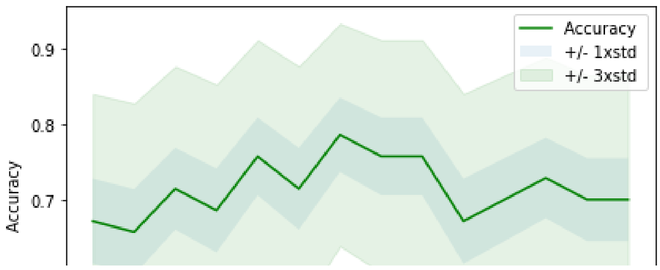
```
#useing a for loop to find the best k
Ks = 15
mean_acc = np.zeros((Ks-1))
std_acc = np.zeros((Ks-1))

for n in range(1,Ks):
    #Train Model and Predict
    neigh = KNeighborsClassifier(n_neighbors
    yhat=neigh.predict(X_test)
    mean_acc[n-1] = metrics.accuracy_score(y_
    std_acc[n-1]=np.std(yhat==y_test)/np.sqrt
mean_acc
```

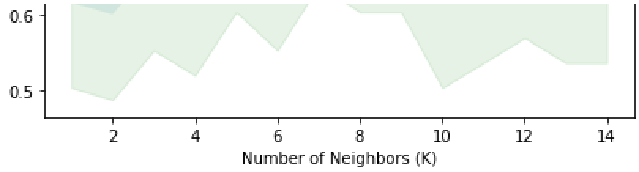
Out[22]:

```
array([0.67142857, 0.65714286, 0.71428571, 0.6
8571429, 0.75714286,
       0.71428571, 0.78571429, 0.75714286, 0.7
5714286, 0.67142857,
       0.7       , 0.72857143, 0.7       , 0.7
])
```

In [23]:

```
#plot the Ks
plt.plot(range(1,Ks),mean_acc,'g')
plt.fill_between(range(1,Ks),mean_acc - 1 * s
plt.fill_between(range(1,Ks),mean_acc - 3 * s
plt.legend(('Accuracy ', '+/- 1xstd','+/- 3xs
plt.ylabel('Accuracy ')
plt.xlabel('Number of Neighbors (K)')
plt.tight_layout()
plt.show()
```





In [24]:

```
#train the model with the best K (7)
neigh = KNeighborsClassifier(n_neighbors = 7)
yhat=neigh.predict(X_test)
yhat[:5]
```

Out[24]:

```
array(['PAIDOFF', 'PAIDOFF', 'PAIDOFF', 'PAIDO
FF', 'PAIDOFF'],
      dtype=object)
```

Decision Tree

In [25]:

```
#load up decision tree and create a predictio
Decision_Tree = DecisionTreeClassifier(criter
Decision_Tree.fit(X_train,y_train)
predTree = Decision_Tree.predict(X_test)
predTree[0:5]
```

Out[25]:

```
array(['COLLECTION', 'COLLECTION', 'PAIDOFF',
'PAIDOFF', 'PAIDOFF'],
      dtype=object)
```

Support Vector Machine

In [26]:

```
#load SVM
clf = svm.SVC(kernel='rbf')
clf.fit(X_train, y_train)
svmYhat = clf.predict(X_test)
svmYhat [0:5]
```

Out[26]:

```
array(['COLLECTION', 'PAIDOFF', 'PAIDOFF', 'PA
IDOFF', 'PAIDOFF'],
      dtype=object)
```

Logistic Regression

All data passed to a dataframe at the end of file

In [27]:

```
LR = LogisticRegression(C=0.01, solver='libli
```

In [28]:

```
LRyhat = LR.predict(X_test)
LRyhat_prob = LR.predict_proba(X_test)
print(LRyhat[:5])
print(LRyhat_prob[:5])
```

```
['COLLECTION' 'PAIDOFF' 'PAIDOFF' 'PAIDOFF' 'P
AIDOFF']
[[0.5034238  0.4965762 ]
 [0.45206111 0.54793889]
 [0.30814132 0.69185868]
 [0.34259428 0.65740572]
 [0.32025894 0.67974106]]
```

Model Evaluation using Test set

First, download and load the test set:

In [29]:

```
!wget -O loan_test.csv https://s3-api.us-geo.
```

```
--2022-06-06 20:06:31--  https://s3-api.us-ge
o.objectstorage.softlayer.net/cf-courses-data/
CognitiveClass/ML0101ENv3/labs/loan_test.csv
Resolving s3-api.us-geo.objectstorage.softlaye
```