Classification with Python

In this notebook we try to practice all the classification algorithms that we 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.

Lets first load required libraries:

In [1]:
```python
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

Lets download the dataset

In [2]:
```
!wget -O loan_train.csv https://cf-courses-data.s3.us.cloud-object-storage.appdomain.c
```
```
--2021-01-15 22:53:41--  https://cf-courses-data.s3.us.cloud-object-storage.appdomain.
cloud/IBMDeveloperSkillsNetwork-ML0101EN-SkillsNetwork/labs/FinalModule_Coursera/data/
loan_train.csv
Resolving cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-data.
s3.us.cloud-object-storage.appdomain.cloud)... 198.23.119.245
Connecting to cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-d
ata.s3.us.cloud-object-storage.appdomain.cloud)|198.23.119.245|: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

2021-01-15 22:53:41 (123 MB/s) - 'loan_train.csv' saved [23101/23101]
```

Load Data From CSV File

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

Out[3]:

	Unnamed: 0	Unnamed: 0.1	loan_status	Principal	terms	effective_date	due_date	age	education	Gender
0	0	0	PAIDOFF	1000	30	9/8/2016	10/7/2016	45	High School or Below	male
1	2	2	PAIDOFF	1000	30	9/8/2016	10/7/2016	33	Bechalor	female
2	3	3	PAIDOFF	1000	15	9/8/2016	9/22/2016	27	college	male
3	4	4	PAIDOFF	1000	30	9/9/2016	10/8/2016	28	college	female
4	6	6	PAIDOFF	1000	30	9/9/2016	10/8/2016	29	college	male

In [4]:
```python
df.shape
```
Out[4]:
```
(346, 10)
```

Convert to date time object

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

Out[5]:

	Unnamed: 0	Unnamed: 0.1	loan_status	Principal	terms	effective_date	due_date	age	education	Gender
0	0	0	PAIDOFF	1000	30	2016-09-08	2016-10-07	45	High School or Below	male
1	2	2	PAIDOFF	1000	30	2016-09-08	2016-10-07	33	Bechalor	female
2	3	3	PAIDOFF	1000	15	2016-09-08	2016-09-22	27	college	male
3	4	4	PAIDOFF	1000	30	2016-09-09	2016-10-08	28	college	female
4	6	6	PAIDOFF	1000	30	2016-09-09	2016-10-08	29	college	male

Data visualization and pre-processing

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

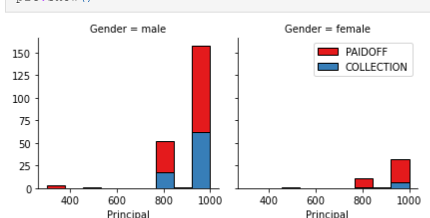
In [6]:
```python
df['loan_status'].value_counts()
```
Out[6]:
```
PAIDOFF       260
COLLECTION     86
Name: loan_status, dtype: int64
```

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

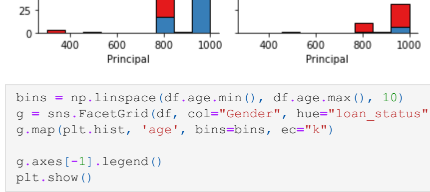
Lets plot some columns to underestand data better:

In [8]:
```python
# notice: installing seaborn might takes a few minutes
!conda install -c anaconda seaborn -y
```
```
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Solving environment: / ^C
failed with repodata from current_repodata.json, will retry with next repodata source.

CondaError: KeyboardInterrupt
```

In [7]:
```python
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()
```



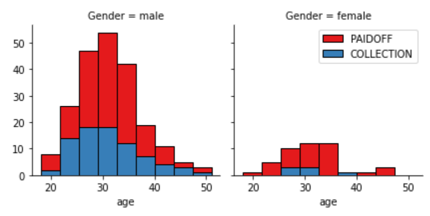
In [8]:
```python
bins = np.linspace(df.age.min(), df.age.max(), 10)
g = sns.FacetGrid(df, col="Gender", hue="loan_status", palette="Set1", col_wrap=2)
g.map(plt.hist, 'age', bins=bins, ec="k")

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



Pre-processing: Feature selection/extraction

Lets look at the day of the week people get the loan

In [9]:
```python
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 dont pay it off, so lets use Feature binarization to set a threshold values less then day 4

In [10]:
```python
df['weekend'] = df['dayofweek'].apply(lambda x: 1 if (x>3)  else 0)
df.head()
```

Out[10]:

	Unnamed: 0	Unnamed: 0.1	loan_status	Principal	terms	effective_date	due_date	age	education	Gender	day
0	0	0	PAIDOFF	1000	30	2016-09-08	2016-10-07	45	High School or Below	male	
1	2	2	PAIDOFF	1000	30	2016-09-08	2016-10-07	33	Bechalor	female	
2	3	3	PAIDOFF	1000	15	2016-09-08	2016-09-22	27	college	male	
3	4	4	PAIDOFF	1000	30	2016-09-09	2016-10-08	28	college	female	
4	6	6	PAIDOFF	1000	30	2016-09-09	2016-10-08	29	college	male	

Convert Categorical features to numerical values

Lets look at gender:

In [11]:
```python
df.groupby(['Gender'])['loan_status'].value_counts(normalize=True)
```
Out[11]:
```
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

Lets convert male to 0 and female to 1:

In [12]:
```python
df['Gender'].replace(to_replace=['male','female'], value=[0,1],inplace=True)
df.head()
```

Out[12]:

	Unnamed: 0	Unnamed: 0.1	loan_status	Principal	terms	effective_date	due_date	age	education	Gender	day
0	0	0	PAIDOFF	1000	30	2016-09-08	2016-10-07	45	High School or Below	0	
1	2	2	PAIDOFF	1000	30	2016-09-08	2016-10-07	33	Bechalor	1	
2	3	3	PAIDOFF	1000	15	2016-09-08	2016-09-22	27	college	0	
3	4	4	PAIDOFF	1000	30	2016-09-09	2016-10-08	28	college	1	
4	6	6	PAIDOFF	1000	30	2016-09-09	2016-10-08	29	college	0	

One Hot Encoding

How about education?

In [13]:
```python
df.groupby(['education'])['loan_status'].value_counts(normalize=True)
```
Out[13]:
```
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
```

Feature befor One Hot Encoding

In [14]:
```python
df[['Principal','terms','age','Gender','education']].head()
```

Out[14]:

	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 [15]:
```python
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()
```

Out[15]:

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

Feature selection

Lets defind feature sets, X:

In [16]:
```python
X = Feature
X[0:5]
```

Out[16]:

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

What are our lables?

In [17]:
```python
y = df['loan_status'].values
y[0:5]
```
Out[17]:
```
array(['PAIDOFF', 'PAIDOFF', 'PAIDOFF', 'PAIDOFF', 'PAIDOFF'],
      dtype=object)
```

Normalize Data

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

In [18]:
```python
X= preprocessing.StandardScaler().fit(X).transform(X)
X[0:5]
```
Out[18]:
```
array([[ 0.51578458,  0.92071769,  2.33152555, -0.42056004, -1.20577805,
        -0.38170062,  1.13639374, -0.86968108],
       [ 0.51578458,  0.92071769,  0.34170148,  2.37778177, -1.20577805,
         2.61985426, -0.87997669, -0.86968108],
       [ 0.51578458, -0.95911111, -0.65321055, -0.42056004, -1.20577805,
        -0.38170062, -0.87997669,  1.14984679],
       [ 0.51578458,  0.92071769, -0.48739188,  2.37778177,  0.82934003,
        -0.38170062, -0.87997669,  1.14984679],
       [ 0.51578458,  0.92071769, -0.3215732 , -0.42056004,  0.82934003,
        -0.38170062, -0.87997669,  1.14984679]])
```

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.

In [19]:
```python
# Import libraries
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import f1_score,log_loss,jaccard_score
```

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 [20]:
```python
# Divido train y test
from sklearn.model_selection import train_test_split
X_trainK, X_testK, y_trainK, y_testK = train_test_split( X, y, test_size=0.2, random_s

# finding 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 = n).fit(X_trainK,y_trainK)
    yhat=neigh.predict(X_testK)
    mean_acc[n-1] = metrics.accuracy_score(y_testK, yhat)
    std_acc[n-1]=np.std(yhat==y_testK)/np.sqrt(yhat.shape[0])
mean_acc

# Better values
maxValue = np.amax(mean_acc)
ind = np.where(mean_acc == maxValue)
indice = ind[0][0] + 1
Kmax = indice

# plot
plt.plot(range(1,Ks),mean_acc,'g')
plt.fill_between(range(1,Ks),mean_acc - 1 * std_acc,mean_acc + 1 * std_acc, alpha=0.1(
plt.fill_between(range(1,Ks),mean_acc - 3 * std_acc,mean_acc + 3 * std_acc, alpha=0.1(
plt.legend(('Accuracy ', '+/- 1xstd','+/- 3xstd'))
plt.ylabel('Accuracy ')
plt.xlabel('Number of Neighbors (K)')
plt.title("Finding the best k")
plt.tight_layout()
plt.show()

print("The best accuracy is %s for k=%s" % (maxValue,Kmax))

# model with the best k
neigh = KNeighborsClassifier(n_neighbors = Kmax).fit(X_trainK,y_trainK)
```


```
The best accuracy is 0.7857142857142857 for k=7
```

Decision Tree

In [21]:
```python
# Train Test Split
X_trainT, X_testT, y_trainT, y_testT = train_test_split(X, y, test_size=0.3, random_st

# Instance
drugTree = DecisionTreeClassifier(criterion="entropy", max_depth = 8) # entropy and de

# Train
drugTree.fit(X_trainT,y_trainT)

# Predict
predTree = drugTree.predict(X_testT)

# Evaluation
print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_testT, predTree))
```
```
DecisionTrees's Accuracy:  0.6634615384615384
```

Support Vector Machine

In [22]:
```python
# Train Test Split
X_trainSVM, X_testSVM, y_trainSVM, y_testSVM = train_test_split(X, y, test_size=0.2, r

# possible kernel functions
kernels = ['rbf','linear','sigmoid','poly']

[fn, maxAcc] = [None,0]

for k in kernels:
    clf = svm.SVC(kernel=k)
    clf.fit(X_trainSVM, y_trainSVM)
    yhatSVM = clf.predict(X_testSVM)
    score = f1_score(y_testSVM, yhatSVM, average='weighted')
    if score > maxAcc:
        [fn, maxAcc] = [k,score]
    print(k,":",f1_score(y_testSVM, yhatSVM, average='weighted') )


print("With F1-score the best solver is %s with accuracy=%s" % (fn,maxAcc))

# best model
clf = svm.SVC(kernel=fn)
clf.fit(X_trainSVM, y_trainSVM)
```
```
rbf : 0.7275882012724117
linear : 0.6914285714285714
sigmoid : 0.6892857142857144
poly : 0.7064793130366899
With F1-score the best solver is rbf with accuracy=0.7275882012724117
```
Out[22]:
```
SVC()
```

Logistic Regression

In [23]:
```python
# Train Test Split
X_trainLR, X_testLR, y_trainLR, y_testLR = train_test_split( X, y, test_size=0.2, rand


# let's try with different values for the regularizations parameter
regularizationRange = np.arange(0.005, 0.1, 0.005)

[bestC,Accuracy] = [0,None]

for c in regularizationRange:
    LR = LogisticRegression(C=c, solver='liblinear').fit(X_trainLR,y_trainLR)
    yhatLR = LR.predict(X_testLR)
    score = f1_score(y_testLR, yhatLR, average='weighted')
    if score > bestC:
        [bestC,Accuracy] = [c,score]

print("The best value of C in the given range is %s with accuracy=%s" % (bestC,Accura

LR = LogisticRegression(C=bestC, solver='liblinear').fit(X_trainLR,y_trainLR)
```
```
The best value of C in the given range is 0.095 with accuracy=0.7048206031256878
```

Model Evaluation using Test set

First, download and load the test set:

In [24]:
```
!wget -O loan_test.csv https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-da
```
```
--2021-01-15 22:55:49--  https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-
data/CognitiveClass/ML0101ENv3/labs/loan_test.csv
Resolving s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-geo.objectstorage.softl
ayer.net)... 67.228.254.196
Connecting to s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-geo.objectstorage.s
oftlayer.net)|67.228.254.196|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3642 (3.6K) [text/csv]
Saving to: 'loan_test.csv'

loan_test.csv       100%[===================>]   3.56K  --.-KB/s    in 0s

2021-01-15 22:55:49 (94.2 MB/s) - 'loan_test.csv' saved [3642/3642]
```

Load Test set for evaluation

In [25]:
```python
test_df = pd.read_csv('loan_test.csv')
test_df.head()
```

Out[25]:

	Unnamed: 0	Unnamed: 0.1	loan_status	Principal	terms	effective_date	due_date	age	education	Gender
0	1	1	PAIDOFF	1000	30	9/8/2016	10/7/2016	50	Bechalor	female
1	5	5	PAIDOFF	300	7	9/9/2016	9/15/2016	35	Master or Above	male
2	21	21	PAIDOFF	1000	30	9/10/2016	10/9/2016	43	High School or Below	female
3	24	24	PAIDOFF	1000	30	9/10/2016	10/9/2016	26	college	male
4	35	35	PAIDOFF	800	15	9/11/2016	9/25/2016	29	Bechalor	male

In [26]:
```python
# Process the dataset

#
test_df['Gender'].replace(to_replace=['male','female'], value=[0,1],inplace=True)

#
test_df['due_date'] = pd.to_datetime(df['due_date'])
test_df['effective_date'] = pd.to_datetime(df['effective_date'])
test_df['dayofweek'] = test_df['effective_date'].dt.dayofweek

#
test_df['weekend'] = test_df['dayofweek'].apply(lambda x: 1 if (x>3)  else 0)

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

#
X_Test = Feature
print ('Test set:', X_Test.shape)

#
X_Test = preprocessing.StandardScaler().fit(X_Test).transform(X_Test)

#
y_Test = test_df['loan_status'].values

#
print ('Test set:', X_Test.shape,  y_Test.shape)
```
```
Test set: (54, 8)
Test set: (54, 8) (54,)
```

In [27]:
```python
""" 1. K-NN """

yhatKNN = neigh.predict(X_Test)

# 1.1. Jaccard
jaccardKNN = jaccard_score(y_Test, yhatKNN,pos_label='PAIDOFF')

# 1.2. F1-score
F1KNN = f1_score(y_Test, yhatKNN, average='weighted')

print(jaccardKNN,F1KNN)


""" 2. Decision Tree """

yhatTREE = drugTree.predict(X_Test)


# 2.1. Jaccard
jaccardT = jaccard_score(y_Test, yhatTREE,pos_label='PAIDOFF')

# 2.2. F1-score
F1T = f1_score(y_Test, yhatTREE, average='weighted')

print(jaccardT,F1T)



""" 3. SVM """

yhatSVM = clf.predict(X_Test)


# 3.1. Jaccard
jaccardSVM = jaccard_score(y_Test, yhatSVM,pos_label='PAIDOFF')

#3.2. F1-score
F1SVM = f1_score(y_Test, yhatSVM, average='weighted')

print(jaccardSVM,F1SVM)


""" 4. Logistic Regression """

# predicciones
yhatLR = LR.predict(X_Test)

# probs de cada clase (acá es paga/no paga; suman 1)
yhat_probLR = LR.predict_proba(X_Test)


# 4.1. Jaccard
jaccardLR = jaccard_score(y_Test, yhatLR,pos_label='PAIDOFF')


# 4.2. F1-score
f1_LR = f1_score(y_Test, yhatLR, average='weighted')


# 4.3. log loss
logLoss = log_loss(y_Test, yhat_probLR)

print(jaccardLR,f1_LR,logLoss)
```
```
0.6730769230769231 0.6453810131971051
0.6136363636363636 0.703520421830281
0.7959183673469388 0.7861952861952862
0.7407407407407407 0.6304176516942475 0.5478960814470889
```

Report

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

Algorithm	Jaccard	F1-score	LogLoss
KNN	0.673	0.645	NA
Decision Tree	0.614	0.704	NA
SVM	0.796	0.786	NA
LogisticRegression	0.741	0.630	0.548

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