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:

Ввод [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

Ввод [2]:

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

```
--2022-02-24 20:12:48--  https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-ML0101EN-SkillsNetwork/labs/FinalModule_Coursera/data/loan_train.csv (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)... 169.63.118.104
Connecting to cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud)|169.63.118.104|: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 0.005s

2022-02-24 20:12:49 (4.25 MB/s) - ‘loan_train.csv’ saved [23101/23101]
```

Load Data From CSV File

Ввод [41]:

```
df = pd.read_csv('loan_train.csv')
render(df.head())
```

Out[41]:

	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

Ввод [18]:

```
df.shape
```

Out[18]: (346, 10)

Convert to date time object

Ввод [42]:

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

Out[42]:

	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 00:00:00	2016-10-07 00:00:00	45	High School or Below	male
1	2	2	PAIDOFF	1000	30	2016-09-08 00:00:00	2016-10-07 00:00:00	33	Bechalor	female
2	3	3	PAIDOFF	1000	15	2016-09-08 00:00:00	2016-09-22 00:00:00	27	college	male
3	4	4	PAIDOFF	1000	30	2016-09-09 00:00:00	2016-10-08 00:00:00	28	college	female
4	6	6	PAIDOFF	1000	30	2016-09-09 00:00:00	2016-10-08 00:00:00	29	college	male

Data visualization and pre-processing

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

Ввод [43]:

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

Out[43]:

```
PAIDOFF       260
COLLECTION     86
Name: loan_status, dtype: int64
```

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

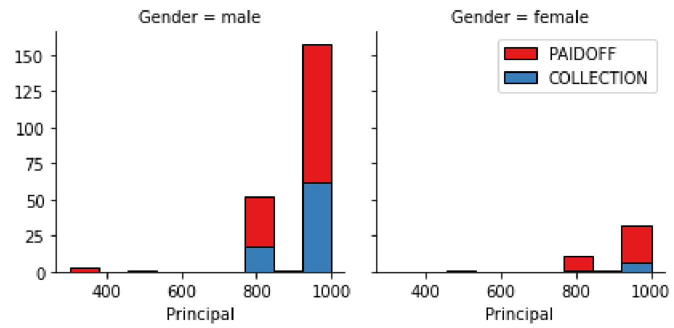
Let's plot some columns to underestand data better:

Ввод [ ]:

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

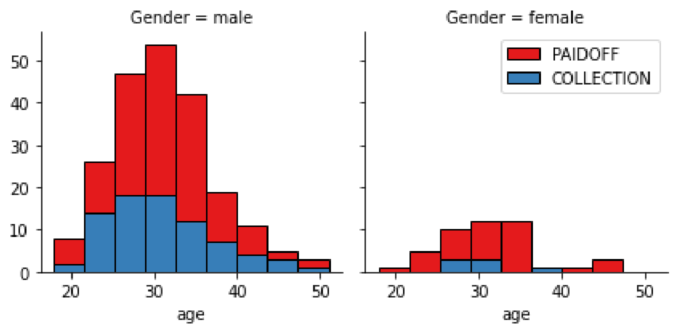
Ввод [21]:

```
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()
```



Ввод [22]:

```
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()
```

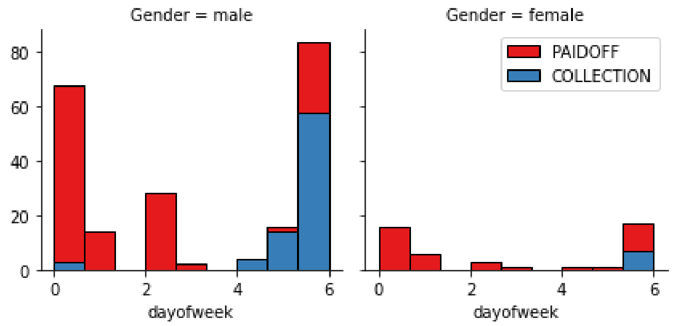


Pre-processing: Feature selection/extraction

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

Ввод [47]:

```
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

Ввод [49]:

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

Out[49]:

	Unnamed: 0	Unnamed: 0.1	loan_status	Principal	terms	effective_date	due_date	age	education	Gender	dayofweek	weekend
0	0	0	PAIDOFF	1000	30	2016-09-08 00:00:00	2016-10-07 00:00:00	45	High School or Below	male	3	0
1	2	2	PAIDOFF	1000	30	2016-09-08 00:00:00	2016-10-07 00:00:00	33	Bechalor	female	3	0
2	3	3	PAIDOFF	1000	15	2016-09-08 00:00:00	2016-09-22 00:00:00	27	college	male	3	0
3	4	4	PAIDOFF	1000	30	2016-09-09 00:00:00	2016-10-08 00:00:00	28	college	female	4	1
4	6	6	PAIDOFF	1000	30	2016-09-09 00:00:00	2016-10-08 00:00:00	29	college	male	4	1

Convert Categorical features to numerical values

Let's look at gender:

Ввод [50]:

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

Out[50]:

```
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:

Ввод [51]:

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

Out[51]:

	Unnamed: 0	Unnamed: 0.1	loan_status	Principal	terms	effective_date	due_date	age	education	Gender	dayofweek	weekend
0	0	0	PAIDOFF	1000	30	2016-09-08 00:00:00	2016-10-07 00:00:00	45	High School or Below	0	3	0
1	2	2	PAIDOFF	1000	30	2016-09-08 00:00:00	2016-10-07 00:00:00	33	Bechalor	1	3	0
2	3	3	PAIDOFF	1000	15	2016-09-08 00:00:00	2016-09-22 00:00:00	27	college	0	3	0
3	4	4	PAIDOFF	1000	30	2016-09-09 00:00:00	2016-10-08 00:00:00	28	college	1	4	1
4	6	6	PAIDOFF	1000	30	2016-09-09 00:00:00	2016-10-08 00:00:00	29	college	0	4	1

One Hot Encoding

How about education?

Ввод [52]:

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

Out[52]:

```
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

Ввод [55]:

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

Out[55]:

	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

Ввод [56]:

```
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)
render(Feature.head())

```

Out[56]:

	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

Let's define feature sets, X:

Ввод [58]:

```
X = Feature
render(X[0:5])
```

Out[58]:

	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?

Ввод [60]:

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

Out[60]:

```
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)

Ввод [61]:

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

Out[61]:

```
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.

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**.

Ввод [62]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4)

print("Train set: ", X_train.shape, y_train.shape)
print("Test set: ", X_test.shape, y_test.shape)
```

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

Ввод [63]:

```
#Training
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
k = 3
#We fit the model:
kNN_model = KNeighborsClassifier(n_neighbors=k).fit(X_train, y_train)
kNN_model
y_pred = kNN_model.predict( X_test )

#Find the max value
accuracies = {}
k_max = 1
acc_max = 0
for k in range(1, 10):
    kNN_model = KNeighborsClassifier(n_neighbors=k).fit(X_train, y_train)
    y_pred = kNN_model.predict( X_test )
    accuracies[k] = accuracy_score(y_test, y_pred)
    print(k, accuracies[k])
```

```
1 0.6714285714285714
2 0.6571428571428571
3 0.7142857142857143
4 0.6857142857142857
5 0.7571428571428571
6 0.7142857142857143
7 0.7857142857142857
8 0.7571428571428571
9 0.7571428571428571
```

Ввод [64]:

```
from sklearn.metrics import f1_score
from sklearn.metrics import jaccard_score
from sklearn import metrics

print("We take k = 7")
knn_model = KNeighborsClassifier(n_neighbors = 7).fit(X_train, y_train)

print("Train set Accuracy (Jaccard): ", metrics.accuracy_score(y_train, knn_model.predict(X_trair
print("Test set Accuracy (Jaccard): ", metrics.accuracy_score(y_test, knn_model.predict(X_test))

print("Train set Accuracy (F1): ", f1_score(y_train, knn_model.predict(X_train), average='weighte
print("Test set Accuracy (F1): ", f1_score(y_test, knn_model.predict(X_test), average='weighted')
```

```
We take k = 7
Train set Accuracy (Jaccard):  0.8079710144927537
Test set Accuracy (Jaccard):  0.7857142857142857
Train set Accuracy (F1):  0.8000194668761034
Test set Accuracy (F1):  0.7766540244416351
```

Decision Tree

Ввод [65]:

```
from sklearn.tree import DecisionTreeClassifier

for d in range(1,10):
    dt = DecisionTreeClassifier(criterion = 'entropy', max_depth = d).fit(X_train, y_train)
    y_pred = dt.predict(X_test)
    print(d, accuracy_score(y_test, y_pred))
```

```
1 0.7857142857142857
2 0.7857142857142857
3 0.6142857142857143
4 0.6142857142857143
5 0.6428571428571429
6 0.7714285714285715
7 0.7571428571428571
8 0.7571428571428571
9 0.6571428571428571
```

Ввод [66]:

```
print("We take depth = 2")
dt = DecisionTreeClassifier(criterion="entropy", max_depth=2).fit(X_train, y_train)

print("Train set Accuracy (Jaccard): ", metrics.accuracy_score(y_train, dt.predict(X_train)))
print("Test set Accuracy (Jaccard): ", metrics.accuracy_score(y_test, dt.predict(X_test)))

print("Train set Accuracy (F1): ", f1_score(y_train, dt.predict(X_train), average='weighted'))
print("Test set Accuracy (F1): ", f1_score(y_test, dt.predict(X_test), average='weighted'))
```

```
We take depth = 2
Train set Accuracy (Jaccard):  0.7427536231884058
Test set Accuracy (Jaccard):  0.7857142857142857
Train set Accuracy (F1):  0.6331163939859591
Test set Accuracy (F1):  0.6914285714285714
```

Support Vector Machine

Ввод [67]:

```
from sklearn import svm
for k in ('linear', 'poly', 'rbf','sigmoid'):
    svm_model = svm.SVC( kernel = k).fit(X_train,y_train)
    svm_yhat = svm_model.predict(X_test)
    print("For kernel: {}, the f1 score is: {}".format(k,f1_score(y_test,svm_yhat, average='weigh
```

```
For kernel: linear, the f1 score is: 0.6914285714285714
For kernel: poly, the f1 score is: 0.7064793130366899
For kernel: rbf, the f1 score is: 0.7275882012724117
For kernel: sigmoid, the f1 score is: 0.6892857142857144
```

Ввод [68]:
```
print("So we choose rbf")
svm_model = svm.SVC( kernel = 'rbf').fit(X_train,y_train)


print("Train set Accuracy (Jaccard): ", metrics.accuracy_score(y_train, svm_model.predict(X_train
print("Test set Accuracy (Jaccard): ", metrics.accuracy_score(y_test, svm_model.predict(X_test)))

print("Train set Accuracy (F1): ", f1_score(y_train, svm_model.predict(X_train), average='weighte
print("Test set Accuracy (F1): ", f1_score(y_test, svm_model.predict(X_test), average='weighted')
```

```
So we choose rbf
Train set Accuracy (Jaccard):  0.782608695652174
Test set Accuracy (Jaccard):  0.7428571428571429
Train set Accuracy (F1):  0.7682165861513688
Test set Accuracy (F1):  0.7275882012724117
```

Logistic Regression

Ввод [85]:
```
from sklearn.metrics import jaccard_score
from sklearn.metrics import f1_score
from sklearn.metrics import log_loss
```

Ввод [112]:
```
test_df['due_date'] = pd.to_datetime(test_df['due_date'])
test_df['effective_date'] = pd.to_datetime(test_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)
test_df['Gender'].replace(to_replace=['male','female'], value=[0,1],inplace=True)

Feature1 = test_df[['Principal','terms','age','Gender','weekend']]
Feature1 = pd.concat([Feature1,pd.get_dummies(test_df['education'])], axis=1)
Feature1.drop(['Master or Above'], axis = 1,inplace=True)


x_loan_test = Feature1
x_loan_test = preprocessing.StandardScaler().fit(x_loan_test).transform(x_loan_test)

y_loan_test = test_df['loan_status'].values
```

Ввод [ ]:
```

```

Model Evaluation using Test set

Ввод [87]:
```
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:

Ввод [73]:
```
!wget -O loan_test.csv https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/Cognitiv
```

```
--2022-02-24 20:32:42--  https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/Cognitive
Class/ML0101ENv3/labs/loan_test.csv (https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-da
ta/CognitiveClass/ML0101ENv3/labs/loan_test.csv)
Resolving s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-geo.objectstorage.softlayer.net)... 6
7.228.254.196
Connecting to s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-geo.objectstorage.softlayer.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

2022-02-24 20:32:43 (19.8 MB/s) - ‘loan_test.csv’ saved [3642/3642]
```

Load Test set for evaluation

Ввод [88]:
```
test_df = pd.read_csv('loan_test.csv')
render(test_df.head())
```

Out[88]:

	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

Ввод [113]:
```
## pre-processing
test_df['due_date'] = pd.to_datetime(test_df['due_date'])
test_df['effective_date'] = pd.to_datetime(test_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)
test_df['Gender'].replace(to_replace=['male','female'], value=[0,1],inplace=True)
test_Feature = test_df[['Principal','terms','age','Gender','weekend']]
test_Feature = pd.concat([test_Feature,pd.get_dummies(test_df['education'])], axis=1)
test_Feature.drop(['Master or Above'], axis = 1,inplace=True)
test_X = preprocessing.StandardScaler().fit(test_Feature).transform(test_Feature)
test_X[0:5]
```

Out[113]:
```
array([[ 0.49362588,  0.92844966,  3.05981865,  1.97714211, -1.30384048,
         2.39791576, -0.79772404, -0.86135677],
       [-3.56269116, -1.70427745,  0.53336288, -0.50578054,  0.76696499,
        -0.41702883, -0.79772404, -0.86135677],
       [ 0.49362588,  0.92844966,  1.88080596,  1.97714211,  0.76696499,
        -0.41702883,  1.25356634, -0.86135677],
       [ 0.49362588,  0.92844966, -0.98251057, -0.50578054,  0.76696499,
        -0.41702883, -0.79772404,  1.16095912],
       [-0.66532184, -0.78854628, -0.47721942, -0.50578054,  0.76696499,
         2.39791576, -0.79772404, -0.86135677]])
```

Ввод [114]:
```
test_y = test_df['loan_status'].values
test_y[0:5]
```

Out[114]:
```
array(['PAIDOFF', 'PAIDOFF', 'PAIDOFF', 'PAIDOFF', 'PAIDOFF'],
      dtype=object)
```

Ввод [120]:
```
from sklearn.linear_model import LinearRegression as lr_model
from sklearn import linear_model

knn_pred = kNN_model.predict(x_loan_test)
j1 = accuracy_score(y_loan_test, knn_pred)

dt_pred = dt.predict(x_loan_test)
j2 = accuracy_score(y_loan_test, dt_pred)

svm_pred = svm_model.predict(x_loan_test)
j3 = accuracy_score(y_loan_test, svm_pred)

lr_pred = lr_model.predict(x_loan_test)
j4 = accuracy_score(y_loan_test, lr_pred)

jaccard = [j1, j2, j3, j4]
jaccard
```

```
[0.7037037037037037,
0.7407407407407407,
0.7962962962962963,
0.7407407407407407]
```

Ввод [123]:

```python
knn_pred = kNN_model.predict(x_loan_test)
f1 = f1_score(y_loan_test, knn_pred, average='weighted')

dt_pred = dt.predict(x_loan_test)
f2 = f1_score(y_loan_test, dt_pred, average='weighted')


svm_pred = svm_model.predict(x_loan_test)
f3 = f1_score(y_loan_test, svm_pred, average='weighted')

lr_pred = lr_model.predict(x_loan_test)
f4 = f1_score(y_loan_test, lr_pred, average='weighted')

f1s = [f1, f2, f3, f4]
f1s
```

```
[0.6736355806123249,
0.6304176516942475,
0.7583503077293734,
0.6604267310789049]
```

Ввод [126]:

```python
from sklearn.metrics import log_loss

lr_pred = lr_model.predict_proba(x_loan_test)
aux = log_loss(y_loan_test, lr_pred)

log_loss = ['NA','NA','NA', aux]
log_loss
```

```
['NA', 'NA', 'NA', 0.5672153379912981]
```

Ввод [ ]:

```python
index   = ["KNN", "Decision Tree", "SVM", "Logistic Regression"]
columns = ["Jaccard", "F1-score", "LogLoss"]

data = [jaccard, f1s, log_loss]
data = np.array(data).T

df = pd.DataFrame(data, index=index, columns=columns)
```

-	Jaccard	F1-score	LogLoss
KNN	0.7037037037037037	0.6736355806123249	NA
Decision Tree	0.7407407407407407	0.6304176516942475	NA
SVM	0.7962962962962963	0.7583503077293734	NA
Logistic Regression	0.7407407407407407	0.6604267310789049	0.5672153379912981

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 (http://cocl.us/ML0101EN-SPSSModeler?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkML0101ENSkillsNetwork20718538-2021-01-01)](http://cocl.us/ML0101EN-SPSSModeler?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkML0101ENSkillsNetwork20718538-2021-01-01)

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 (https://cocl.us/ML0101EN_DSX?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkML0101ENSkillsNetwork20718538-2021-01-01)

Thanks for completing this lesson!

Author: Saeed Aghabozorgi (https://ca.linkedin.com/in/saeedaghabozorgi?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkML0101ENSkillsNetwork20718538-2021-01-01?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkML0101ENSkillsNetwork20718538-2021-01-01)

Saeed Aghabozorgi (https://ca.linkedin.com/in/saeedaghabozorgi), 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