KNN

November 1, 2021

KNN Metrics

- L1 Norm is a calculation of the Manhattan distance from the origin of the vector space. $\|v\|_1 = |a_1| + |a_2| + |a_3|$
- L2 norm calculates the distance of the vector coordinate from the origin of the vector space.
  $\|v\|_2 = \text{sqrt}(a_1^2 + a_2^2 + a_3^2)$

	Attributes		
Sample	Age	Income	Marital Status
S01	33	5432	Single
S01	59	5439	Married
S01	55	4211	Widowed
S01	37	3711	Divorced

L1 norm

- Calculate the L1 norm between Lisa (31) and the samples if she is single and earns 4738.
  $\|v\|_1 = |a_1| + |a_2| + |a_3|$
  1. (Single=0, Married=1, Widowed=2, Divorced=3)
  2. $Lisa\_S01 = |\text{31-33}| + |\text{4738-5432}| + |\text{single-single}| = 696$
  3. $Lisa\_S02 = |\text{31-59}| + |\text{4738-5439}| + |\text{single-Married}| = 730$
  4. $Lisa\_S03 = |\text{31-55}| + |\text{4738-4211}| + |\text{single-Widowed}| = 553$
  5. $Lisa\_S04 = |\text{31-37}| + |\text{4738-3711}| + |\text{single-Divorced}| = 1035$

L2 norm

- Calculate the L2 norm between Maggie (38) and the samples if she is married and has earnings of 6739.
  $\|v\|_2 =$ sqrt$(a_1^2 + a_2^2 + a_3^2\ )$

  1. (Single=0, Married=1, Widowed=2, Divorced=3)
  2. $Maggie\_S01 =$ sqrt$((38-33)^2 + (6739-5432)^2 + (Married - single)^2)$= 1307.01
  3. $Maggie\_S02 =$ sqrt$((38-59)^2 + (6739-5439)^2 + (Married - Married)^2)$= 1300.17
  4. $Maggie_S03 =$sqrt$(\ (38-55)^2 + (6739-4211)^2 + (Married - Widowed)^2) = 2528.06$
  5. $Maggie\_S04 =$ sqrt$((38-37)^2 + (6739-3711)^2 + (Married - Divorced)^2) = 3028.00$

Correlation between attributes and target(class)

- Pearson's Correlation Coefficient: f_regression()
- ANOVA: f_classif()
- Chi-Squared: chi2()
- Mutual Information: mutual_info_classif() and mutual_info_regression()

  All the listed functions are found in the scikit-learn library

Classification with KNN

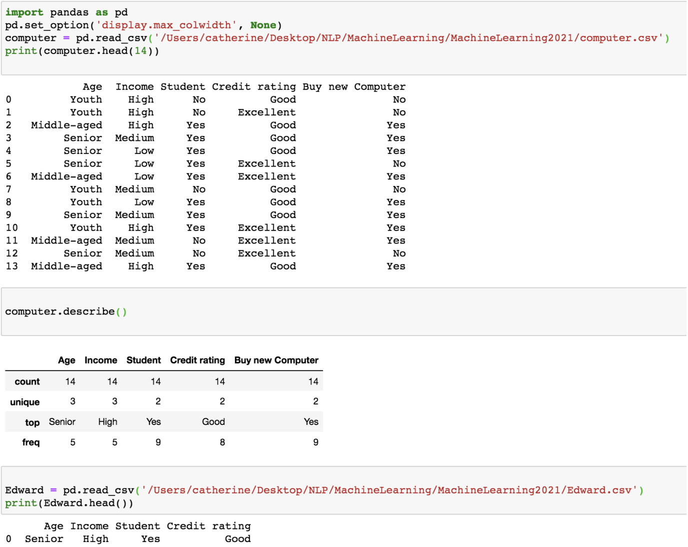
Sample	Attributes				Buy new
	Age	Income	Student	Credit rating	Computer
S01	Youth	High	No	Good	No
S02	Youth	High	No	Excellent	No
S03	Middle-aged	High	Yes	Good	Yes
S04	Senior	Medium	Yes	Good	Yes
S05	Senior	Low	Yes	Good	Yes
S06	Senior	Low	Yes	Excellent	No
S07	Middle-aged	Low	Yes	Excellent	Yes
S08	Youth	Medium	No	Good	No
S09	Youth	Low	Yes	Good	Yes
S10	Senior	Medium	Yes	Good	Yes
S11	Youth	High	Yes	Excellent	Yes
S12	Middle-aged	Medium	No	Excellent	Yes
S13	Senior	Medium	No	Excellent	No
S14	Middle-aged	High	Yes	Good	Yes

Classification with KNN

Decide if Edward, the senior student, would buy a new PC with his high income and good credit rating.

Sample	Attributes				Buy new
	Age	Income	Student	Credit rating	Computer
Edward	Senior	High	Yes	Good	?

Classification with KNN



```python
import pandas as pd
pd.set_option('display.max_colwidth', None)
computer = pd.read_csv('/Users/catherine/Desktop/NLP/MachineLearning/MachineLearning2021/computer.csv')
print(computer.head(14))
```

```
            Age  Income Student Credit rating Buy new Computer
0         Youth    High      No          Good               No
1         Youth    High      No     Excellent               No
2   Middle-aged    High     Yes          Good              Yes
3        Senior  Medium     Yes          Good              Yes
4        Senior     Low     Yes          Good              Yes
5        Senior     Low     Yes     Excellent               No
6   Middle-aged     Low     Yes     Excellent              Yes
7         Youth  Medium      No          Good               No
8         Youth     Low     Yes          Good              Yes
9        Senior  Medium     Yes          Good              Yes
10        Youth    High     Yes     Excellent              Yes
11  Middle-aged  Medium      No     Excellent              Yes
12       Senior  Medium      No     Excellent               No
13  Middle-aged    High     Yes          Good              Yes
```

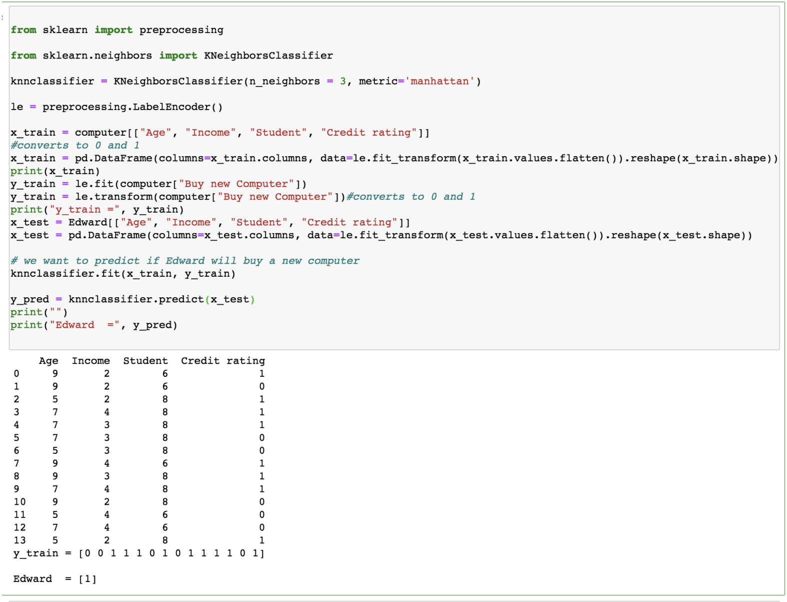
```python
computer.describe()
```

	Age	Income	Student	Credit rating	Buy new Computer
count	14	14	14	14	14
unique	3	3	2	2	2
top	Senior	High	Yes	Good	Yes
freq	5	5	9	8	9

```python
Edward = pd.read_csv('/Users/catherine/Desktop/NLP/MachineLearning/MachineLearning2021/Edward.csv')
print(Edward.head())
```

```
      Age Income Student Credit rating
0  Senior   High     Yes          Good
```

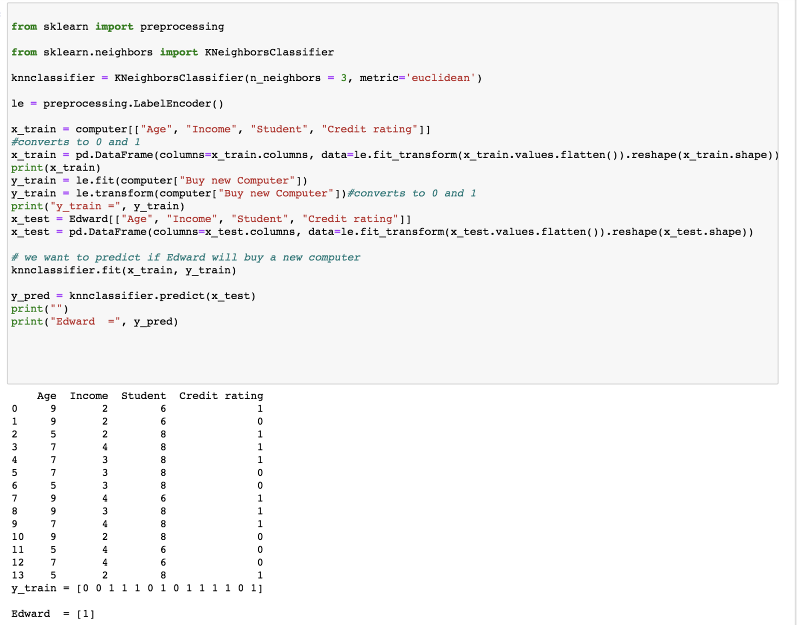
Classification with KNN (metric='manhattan')



```python
from sklearn import preprocessing

from sklearn.neighbors import KNeighborsClassifier

knnclassifier = KNeighborsClassifier(n_neighbors = 3, metric='manhattan')

le = preprocessing.LabelEncoder()

x_train = computer[["Age", "Income", "Student", "Credit rating"]]
#converts to 0 and 1
x_train = pd.DataFrame(columns=x_train.columns, data=le.fit_transform(x_train.values.flatten()).reshape(x_train.shape))
print(x_train)
y_train = le.fit(computer["Buy new Computer"])
y_train = le.transform(computer["Buy new Computer"])#converts to 0 and 1
print("y_train =", y_train)
x_test = Edward[["Age", "Income", "Student", "Credit rating"]]
x_test = pd.DataFrame(columns=x_test.columns, data=le.fit_transform(x_test.values.flatten()).reshape(x_test.shape))

# we want to predict if Edward will buy a new computer
knnclassifier.fit(x_train, y_train)

y_pred = knnclassifier.predict(x_test)
print("")
print("Edward  =", y_pred)
```

```
    Age  Income  Student  Credit rating
0     9       2        6              1
1     9       2        6              0
2     5       2        8              1
3     7       4        8              1
4     7       3        8              1
5     7       3        8              0
6     5       3        8              0
7     9       4        6              1
8     9       3        8              1
9     7       4        8              1
10    9       2        8              0
11    5       4        6              0
12    7       4        6              0
13    5       2        8              1
y_train = [0 0 1 1 1 0 1 0 1 1 1 1 0 1]

Edward  = [1]
```

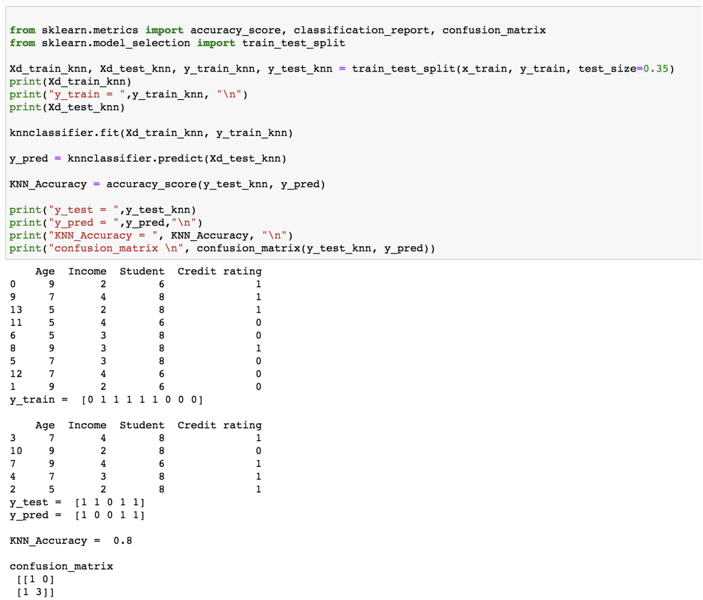
Classification with KNN (metric='euclidean')



```python
from sklearn import preprocessing

from sklearn.neighbors import KNeighborsClassifier

knnclassifier = KNeighborsClassifier(n_neighbors = 3, metric='euclidean')

le = preprocessing.LabelEncoder()

x_train = computer[["Age", "Income", "Student", "Credit rating"]]
#converts to 0 and 1
x_train = pd.DataFrame(columns=x_train.columns, data=le.fit_transform(x_train.values.flatten()).reshape(x_train.shape))
print(x_train)
y_train = le.fit(computer["Buy new Computer"])
y_train = le.transform(computer["Buy new Computer"])#converts to 0 and 1
print("y_train =", y_train)
x_test = Edward[["Age", "Income", "Student", "Credit rating"]]
x_test = pd.DataFrame(columns=x_test.columns, data=le.fit_transform(x_test.values.flatten()).reshape(x_test.shape))

# we want to predict if Edward will buy a new computer
knnclassifier.fit(x_train, y_train)

y_pred = knnclassifier.predict(x_test)
print("")
print("Edward  =", y_pred)
```

```
    Age  Income  Student  Credit rating
0     9       2        6              1
1     9       2        6              0
2     5       2        8              1
3     7       4        8              1
4     7       3        8              1
5     7       3        8              0
6     5       3        8              0
7     9       4        6              1
8     9       3        8              1
9     7       4        8              1
10    9       2        8              0
11    5       4        6              0
12    7       4        6              0
13    5       2        8              1
y_train = [0 0 1 1 1 0 1 0 1 1 1 1 0 1]

Edward  = [1]
```

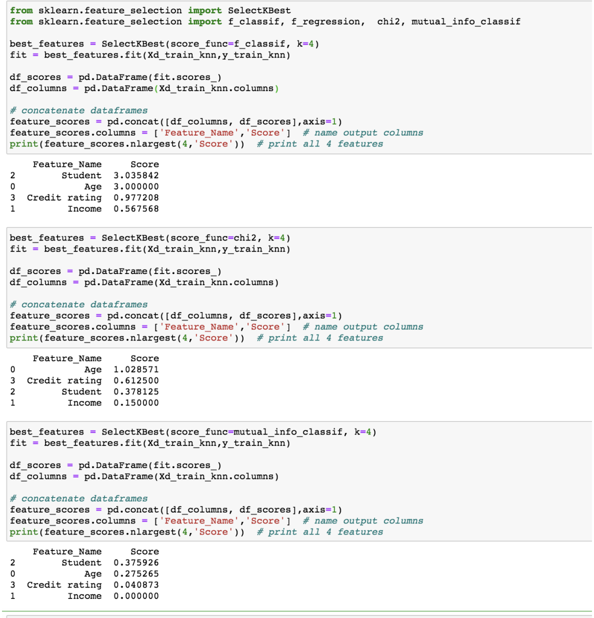
KNN performance



```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.model_selection import train_test_split

Xd_train_knn, Xd_test_knn, y_train_knn, y_test_knn = train_test_split(x_train, y_train, test_size=0.35)
print(Xd_train_knn)
print("y_train = ",y_train_knn, "\n")
print(Xd_test_knn)

knnclassifier.fit(Xd_train_knn, y_train_knn)

y_pred = knnclassifier.predict(Xd_test_knn)

KNN_Accuracy = accuracy_score(y_test_knn, y_pred)

print("y_test = ",y_test_knn)
print("y_pred = ",y_pred,"\n")
print("KNN_Accuracy = ", KNN_Accuracy, "\n")
print("confusion_matrix \n", confusion_matrix(y_test_knn, y_pred))
```

```
    Age  Income  Student  Credit rating
0     9       2        6              1
9     7       4        8              1
13    5       2        8              1
11    5       4        6              0
6     5       3        8              0
8     9       3        8              1
5     7       3        8              0
12    7       4        6              0
1     9       2        6              0
y_train =  [0 1 1 1 1 1 0 0 0]

    Age  Income  Student  Credit rating
3     7       4        8              1
10    9       2        8              0
7     9       4        6              1
4     7       3        8              1
2     5       2        8              1
y_test =  [1 1 0 1 1]
y_pred =  [1 0 0 1 1]

KNN_Accuracy =  0.8

confusion_matrix
 [[1 0]
 [1 3]]
```

Feature selection



```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif, f_regression,  chi2, mutual_info_classif

best_features = SelectKBest(score_func=f_classif, k=4)
fit = best_features.fit(Xd_train_knn,y_train_knn)

df_scores = pd.DataFrame(fit.scores_)
df_columns = pd.DataFrame(Xd_train_knn.columns)

# concatenate dataframes
feature_scores = pd.concat([df_columns, df_scores],axis=1)
feature_scores.columns = ['Feature_Name','Score']  # name output columns
print(feature_scores.nlargest(4,'Score'))  # print all 4 features
```

```
    Feature_Name     Score
2        Student  3.035842
0            Age  3.000000
3  Credit rating  0.977208
1         Income  0.567568
```

```
best_features = SelectKBest(score_func=chi2, k=4)
fit = best_features.fit(Xd_train_knn,y_train_knn)

df_scores = pd.DataFrame(fit.scores_)
df_columns = pd.DataFrame(Xd_train_knn.columns)

# concatenate dataframes
feature_scores = pd.concat([df_columns, df_scores],axis=1)
feature_scores.columns = ['Feature_Name','Score']  # name output columns
print(feature_scores.nlargest(4,'Score'))  # print all 4 features
```

```
    Feature_Name     Score
0            Age  1.028571
3  Credit rating  0.612500
2        Student  0.378125
1         Income  0.150000
```

```
best_features = SelectKBest(score_func=mutual_info_classif, k=4)
fit = best_features.fit(Xd_train_knn,y_train_knn)

df_scores = pd.DataFrame(fit.scores_)
df_columns = pd.DataFrame(Xd_train_knn.columns)

# concatenate dataframes
feature_scores = pd.concat([df_columns, df_scores],axis=1)
feature_scores.columns = ['Feature_Name','Score']  # name output columns
print(feature_scores.nlargest(4,'Score'))  # print all 4 features
```

```
    Feature_Name     Score
2        Student  0.375926
0            Age  0.275265
3  Credit rating  0.040873
1         Income  0.000000
```