

## k\_neighbors\_\_final

January 24, 2023

```
[99]: import numpy as np
import pandas as pd
import matplotlib as plt
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.multiclass import OneVsRestClassifier
from sklearn.metrics import precision_recall_curve, roc_curve, accuracy_score, \
    precision_score, recall_score
from sklearn import datasets, neighbors
from mlxtend.plotting import plot_decision_regions
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.naive_bayes import GaussianNB
```

```
[100]: df_userdata = pd.read_csv('final_userdata_min_40.csv')
df_userdata = df_userdata.drop(df_userdata.columns[0:11],axis =1 )
df_userdata = df_userdata.drop('liked_recipes',axis =1 )
df_userdata.kitchen.unique()
```

```
[100]: array(['frans', 'aziatisch', 'hollands', 'mexicaans', 'italiaans',
        'mediterraan', 'amerikaans'], dtype=object)
```

```
[101]: for keuken in range(len(df_userdata)):

    if df_userdata['kitchen'].iloc[keuken] == 'aziatisch':
        df_userdata['kitchen'].iloc[keuken] = 0
    elif df_userdata['kitchen'].iloc[keuken] == 'frans':
        df_userdata['kitchen'].iloc[keuken] = 1
    elif df_userdata['kitchen'].iloc[keuken] == 'hollands':
        df_userdata['kitchen'].iloc[keuken] = 2
    elif df_userdata['kitchen'].iloc[keuken] == 'italiaans':
        df_userdata['kitchen'].iloc[keuken] = 3
    elif df_userdata['kitchen'].iloc[keuken] == 'mexicaans':
        df_userdata['kitchen'].iloc[keuken] = 4
```

```

elif df_userdata['kitchen'].iloc[keuken] == 'mediterraan':
    df_userdata['kitchen'].iloc[keuken] = 5
elif df_userdata['kitchen'].iloc[keuken] == 'amerikaans':
    df_userdata['kitchen'].iloc[keuken] = 6

```

```
df_userdata
```

/tmp/ipykernel\_43015/3680158733.py:6: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
df_userdata['kitchen'].iloc[keuken] = 1
```

/tmp/ipykernel\_43015/3680158733.py:4: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
df_userdata['kitchen'].iloc[keuken] = 0
```

/tmp/ipykernel\_43015/3680158733.py:8: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
df_userdata['kitchen'].iloc[keuken] = 2
```

/tmp/ipykernel\_43015/3680158733.py:12: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
df_userdata['kitchen'].iloc[keuken] = 4
```

/tmp/ipykernel\_43015/3680158733.py:10: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
df_userdata['kitchen'].iloc[keuken] = 3
```

/tmp/ipykernel\_43015/3680158733.py:14: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
df_userdata['kitchen'].iloc[keuken] = 5
```

/tmp/ipykernel\_43015/3680158733.py:16: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: <https://pandas.pydata.org/pandas->

```
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
df_userdata['kitchen'].iloc[keuken] = 6
```

```
[101]:      chilivlokken  gemalen kaneel  sjalot  knoflook  rode peper  zout  \
0                0                0        0          1          0      0
1                0                0        1          1          0      1
2                0                0        1          1          0      0
3                0                0        1          1          0      0
4                0                0        1          1          0      0
...
1400             1                1        1          1          0      1
1401             0                0        1          1          0      0
1402             0                0        0          1          0      1
1403             0                0        1          1          1      0
1404             0                0        0          0          1      0

      vastkokende aardappel  zoete aardappel  vissaus  middelgrote ui  ...  \
0                0                0          0                0      ...
1                1                0          0                0      ...
2                0                0          0                1      ...
3                0                1          0                0      ...
4                1                0          0                1      ...
...
1400             0                0          0                0      ...
1401             0                1          0                0      ...
1402             1                0          0                0      ...
1403             0                0          0                0      ...
1404             1                0          0                0      ...

      (arachide)olie  tomatenblokjes  zoete puntpaprika  (olijf)olie  \
0                0                0                0                0
1                0                0                0                0
2                0                0                0                0
3                0                1                0                0
4                0                0                0                0
...
1400             0                0                0                0
1401             0                0                1                1
1402             1                1                0                0
1403             0                0                0                0
1404             0                0                0                0

      kruimige aardappelen  magere gerookte spekreepjes  takje rozemarijn  \
0                1                0                0
1                0                0                0
2                0                0                1
3                0                0                1
```

4		0		0		0
...		...		...		...
1400		0		0		0
1401		1		0		0
1402		0		0		0
1403		0		0		1
1404		0		0		0

	boter	aardappelen	kitchen
0	0	1	1
1	1	0	1
2	1	0	1
3	0	0	1
4	0	0	1
...	...	...	...
1400	0	0	1
1401	0	0	2
1402	0	0	1
1403	0	0	0
1404	0	0	0

[1405 rows x 172 columns]

```
[102]: y= df_userdata['kitchen']
X = df_userdata.drop('kitchen', axis = 1)

y=y.astype('int')

X_main, X_test, y_main, y_test = train_test_split(X, y, test_size = 0.2,
↳random_state=42, stratify=y)
X_train, X_val, y_train, y_val = train_test_split(X_main, y_main, test_size = 0.
↳25, random_state=42, stratify=y_main)
X_test
```

```
[102]: chilivlokken  gemalen kaneel  sjalot  knoflook  rode peper  zout  \
1346           0           0           0           1           1           0
1281           0           0           1           1           0           0
1324           0           0           0           1           1           1
853            0           0           0           1           0           0
981            0           0           1           1           1           0
...           ...           ...           ...           ...           ...
602            0           0           0           1           0           0
1140           0           0           0           1           0           1
1266           0           0           0           1           0           0
418            0           0           1           1           0           0
901            0           0           0           1           1           1
```

	vastkokende aardappel	zoete aardappel	vissaus	middelgrote ui	...	\
1346	0	1	0	0	...	
1281	0	0	0	0	...	
1324	0	0	0	1	...	
853	0	0	0	0	...	
981	0	0	0	0	...	
...	...	...	...	...	...	
602	0	0	0	0	...	
1140	0	0	0	0	...	
1266	0	0	0	0	...	
418	0	1	0	1	...	
901	0	0	0	1	...	

	kippenbouillon van tablet	(arachide)olie	tomatenblokjes	\
1346	0	0	0	
1281	0	0	0	
1324	0	0	0	
853	1	0	0	
981	0	0	0	
...	...	...	...	
602	0	1	1	
1140	0	0	1	
1266	0	0	0	
418	0	0	0	
901	0	0	1	

	zoete puntpaprika	(olijf)olie	kruimige aardappelen	\
1346	0	0	0	
1281	0	0	0	
1324	0	0	0	
853	0	0	0	
981	0	0	0	
...	...	...	...	
602	0	0	0	
1140	0	0	0	
1266	0	0	0	
418	0	0	1	
901	0	1	0	

	magere gerookte spekreepjes	takje rozemarijn	boter	aardappelen
1346	0	0	0	0
1281	0	0	0	0
1324	0	0	0	0
853	0	0	0	0
981	1	0	0	0
...	...	...	...	...
602	0	0	0	0

1140	0	1	0	0
1266	0	0	0	0
418	0	0	0	0
901	1	0	0	0

[281 rows x 171 columns]

```
[278]: model = RandomForestClassifier()
model.fit(X_train,y_train)
y_proba = model.predict(X_val)
```

```
[279]: print('Acc_score',accuracy_score(y_proba,y_val))
print('prec_score',precision_score(y_proba,y_val, average = 'macro'))
print('rec_score',recall_score(y_proba,y_val, average = 'macro'))
```

```
Acc_score 0.9644128113879004
prec_score 0.9644599303135888
rec_score 0.9649198332491933
```

```
[280]: #params = { 'criterion' : ['gini', 'entropy', 'log_loss'],
#              'max_depth' : [3,5,8,10,15,20,25,30,40,50,60,70,80,90,100],
#              'min_samples_split' : [5,6,7,8,9,10,20,30,40,50,60]
#              }
```

```
[281]: #params = { 'leaf_size' : [1,2,3,4,5,6,7,8,9,10,20,30,40,50,60,70,80,90],
#              'n_neighbors' : [1,2,3,4,5,6,7,8,9,10],
#              'weights' : ['uniform', 'distance'],
#              'algorithm' : ['auto', 'ball_tree', 'kd_tree', 'brute']
#              }
```

```
[286]: params = { 'n_estimators' : [10,20,30,40,50,80,100,130,150,200,250,300,350,400],
                  'criterion' : ['gini', 'entropy', 'log_loss'],
                  'max_depth' : [3,5,8,10,15,20,25,30,40,50,60,70,80,90,100],
                  'min_samples_split' : [5,6,7,8,9,10,20,30,40,50,60]
                  }
```

```
[287]: randomsearch = RandomizedSearchCV(estimator= model ,param_distributions=␣
    ↪params, n_iter=30, cv=20, random_state=42)
```

```
[288]: search = randomsearch.fit(X_train,y_train)
search.best_estimator_
```

```
[288]: RandomForestClassifier(criterion='entropy', max_depth=20, min_samples_split=5,
                             n_estimators=350)
```

```
[289]: model.score(X_val,y_val)
```

[289]: 0.9644128113879004

```
[293]: #model = KNeighborsClassifier(algorithm='brute', leaf_size=1, n_neighbors=8,
      ↪weights='distance')
      #model = DecisionTreeClassifier(max_depth=25, min_samples_split=20)
      #model = MultinomialNB()
      RandomForestClassifier(criterion='entropy', max_depth=20, min_samples_split=5,
      ↪n_estimators=350)
```

[293]: RandomForestClassifier(criterion='entropy', max\_depth=20, min\_samples\_split=5,
n\_estimators=350)

```
[294]: model.fit(X_train,y_train)
      y_proba = model.predict(X_test)
```

```
[295]: print('Acc_score',accuracy_score(y_proba,y_test))
      print('prec_score',precision_score(y_proba,y_test, average = 'macro'))
      print('rec_score',recall_score(y_proba,y_test, average = 'macro'))
```

Acc\_score 0.9786476868327402  
prec\_score 0.9785714285714285  
rec\_score 0.9791568754557977

```
[296]: from sklearn.metrics import confusion_matrix, classification_report

      print(confusion_matrix(y_test, y_proba))
      print(classification_report(y_test, y_proba))
```

```
[[41  0  0  0  0  0  0]
 [ 1 39  0  0  0  0  0]
 [ 1  1 38  0  0  0  0]
 [ 0  0  0 39  0  1  0]
 [ 0  0  0  0 40  0  0]
 [ 0  0  0  1  0 39  0]
 [ 0  0  0  0  1  0 39]]
```

	precision	recall	f1-score	support
0	0.95	1.00	0.98	41
1	0.97	0.97	0.97	40
2	1.00	0.95	0.97	40
3	0.97	0.97	0.97	40
4	0.98	1.00	0.99	40
5	0.97	0.97	0.97	40
6	1.00	0.97	0.99	40
accuracy			0.98	281
macro avg	0.98	0.98	0.98	281

weighted avg	0.98	0.98	0.98	281
--------------	------	------	------	-----

```
[271]: model.score(X_test,y_test)
```

```
[271]: 0.8540925266903915
```

```
[272]: from sklearn.utils.multiclass import unique_labels
unique_labels(y_test)
```

```
[272]: array([0, 1, 2, 3, 4, 5, 6])
```

```
[273]: def plot(Y_test, y_proba):
labels= unique_labels(y_test)
columns=[f"Predicted {label}" for label in labels]
index=[f"Actual {label}" for label in labels]
table= pd.DataFrame(confusion_matrix(y_test, y_proba), columns=columns,
↪index=index)
return table
```

```
[274]: plot(y_test, y_proba)
```

```
[274]:
```

	Predicted 0	Predicted 1	Predicted 2	Predicted 3	Predicted 4	\
Actual 0	39	2	0	0	0	
Actual 1	0	36	0	4	0	
Actual 2	2	4	33	1	0	
Actual 3	0	1	0	34	0	
Actual 4	0	0	0	0	39	
Actual 5	0	0	1	4	4	
Actual 6	1	1	0	0	6	

	Predicted 5	Predicted 6
Actual 0	0	0
Actual 1	0	0
Actual 2	0	0
Actual 3	5	0
Actual 4	0	1
Actual 5	28	3
Actual 6	1	31

```
[275]: import seaborn as sns
```

```
[276]: def plot2(Y_test, y_proba):
labels= unique_labels(y_test)
columns=[f"Predicted {label}" for label in labels]
index=[f"Actual {label}" for label in labels]
```



```

table= pd.DataFrame(confusion_matrix(y_test, y_proba), columns=columns,
↪index=index)
heatmap=sns.heatmap(table, annot=True, fmt="d", cmap= "Blues")
heatmap.set_xlabel('Predicted labels');
heatmap.set_ylabel('True labels');
heatmap.set_title('Confusion Matrix KNearest Neighbours');
heatmap.xaxis.set_ticklabels(["Aziatisch", "Frans", "Hollands",
↪"Italiaans", "Mexicaans", "Mediterraan", "Amerikaans"]);
heatmap.yaxis.set_ticklabels(["Aziatisch", "Frans", "Hollands",
↪"Italiaans", "Mexicaans", "Mediterraan", "Amerikaans"], rotation=0);
return heatmap

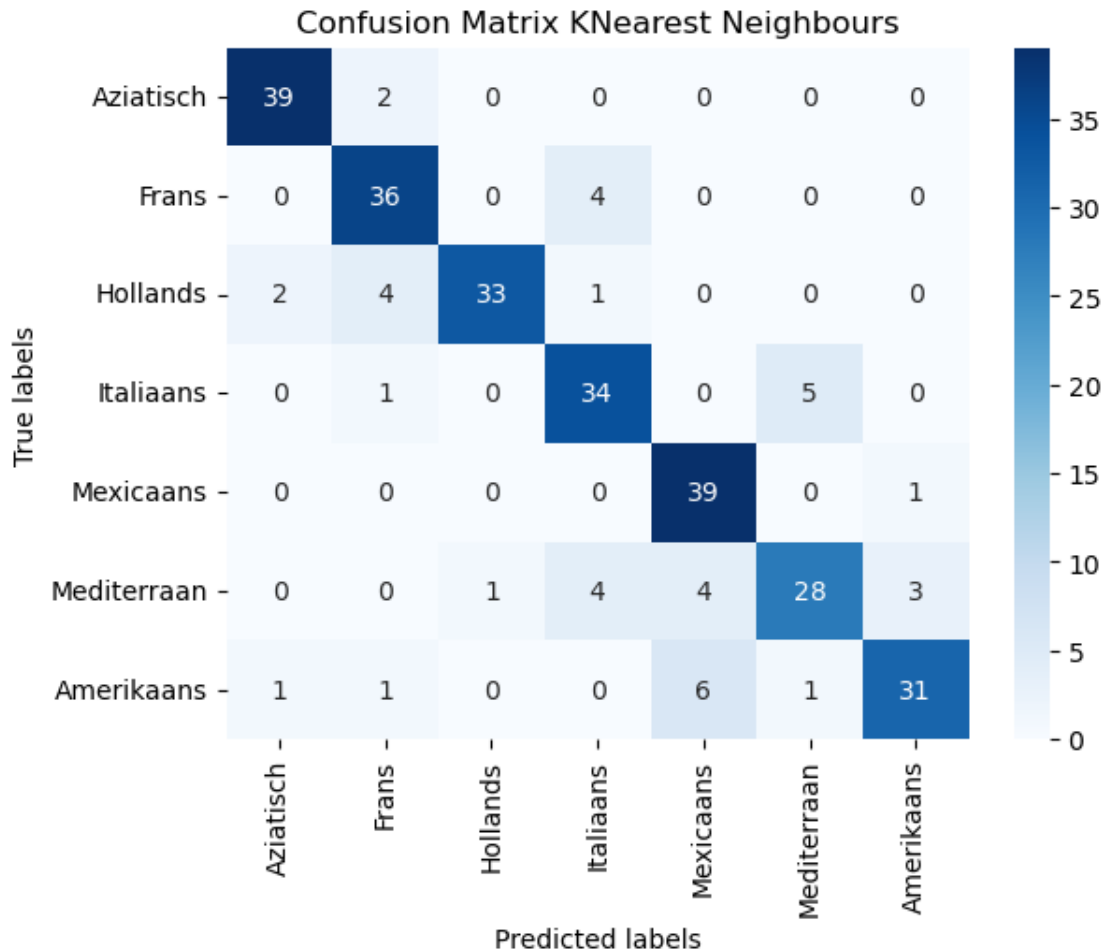
```

```
[277]: plot2(y_test, y_proba)
```

```

[277]: <AxesSubplot:title={'center':'Confusion Matrix KNearest Neighbours'},
xlabel='Predicted labels', ylabel='True labels'>

```



```
[340]: if df_userdata['kitchen'].iloc[keuken] == 'aziatisch':  
        df_userdata['kitchen'].iloc[keuken] = 0  
    elif df_userdata['kitchen'].iloc[keuken] == 'frans':  
        df_userdata['kitchen'].iloc[keuken] = 1  
    elif df_userdata['kitchen'].iloc[keuken] == 'hollands':  
        df_userdata['kitchen'].iloc[keuken] = 2  
    elif df_userdata['kitchen'].iloc[keuken] == 'italiaans':  
        df_userdata['kitchen'].iloc[keuken] = 3  
    elif df_userdata['kitchen'].iloc[keuken] == 'mexicaans':  
        df_userdata['kitchen'].iloc[keuken] = 4  
    elif df_userdata['kitchen'].iloc[keuken] == 'mediterraan':  
        df_userdata['kitchen'].iloc[keuken] = 5
```