03b_LAB_KNN

May 15, 2022

1 Machine Learning Foundation

1.1 Course 3, Part b: K-Nearest Neighbor LAB

1.2 Introduction and Learning Goals

In this lab, we will explore classification using the K-Nearest Neighbors approach. We will use a customer churn dataset from the telecom industry, which includes customer data such as long-distance usage, data usage, monthly revenue, types of offerings, and other services purchased by customers. The data, based on a fictional telecom firm, includes several Excel files which have been combined and are available in the course materials. We are using the subset of customers who have phone accounts. Since the data includes a mix of numeric, categorical, and ordinal variables, we will load this data and do some preprocessing. Then we will use K-nearest neighbors to predict customer churn rates.

After completing this lab, you should have a working understanding of how to preprocess a variety of variables to apply the K-Nearest Neighbors algorithm, understand how to choose K, and understand how to evaluate model performance.

[1]:
```
def warn(*args, **kwargs):
    pass
import warnings
warnings.warn = warn

import pandas as pd, numpy as np, matplotlib.pyplot as plt, os, sys, seaborn as␣
 ↪sns
```

1.3 Question 1

- We begin by importing the data. Examine the columns and data.
- Notice that the data contains a unique ID, an indicator for phone customer status, total lifetime value, total revenue, and a bank-estimated churn score. We will not be using these features, so they can be dropped from the data.
- Begin by taking an initial look at the data, including both numeric and non-numeric features.

[2]:
```
### BEGIN SOLUTION

df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.
 ↪cloud/IBM-ML241EN-SkillsNetwork/labs/datasets/churndata_processed.csv")
```

[3]:
```
round(df.describe(),2)
```

[3]:

	months	multiple	gb_mon	security	backup	protection	support	\
count	7043.00	7043.00	7043.00	7043.00	7043.00	7043.00	7043.00	
mean	0.43	0.42	0.24	0.29	0.34	0.34	0.29	
std	0.40	0.49	0.24	0.45	0.48	0.48	0.45	
min	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
25%	0.00	0.00	0.04	0.00	0.00	0.00	0.00	
50%	0.25	0.00	0.20	0.00	0.00	0.00	0.00	
75%	0.75	1.00	0.32	1.00	1.00	1.00	1.00	
max	1.00	1.00	1.00	1.00	1.00	1.00	1.00	

	unlimited	contract	paperless	…	payment_Credit Card	\
count	7043.00	7043.00	7043.00	…	7043.00	
mean	0.67	0.38	0.59	…	0.39	
std	0.47	0.42	0.49	…	0.49	
min	0.00	0.00	0.00	…	0.00	
25%	0.00	0.00	0.00	…	0.00	
50%	1.00	0.00	1.00	…	0.00	
75%	1.00	1.00	1.00	…	1.00	
max	1.00	1.00	1.00	…	1.00	

	payment_Mailed Check	internet_type_DSL	internet_type_Fiber Optic	\
count	7043.00	7043.00	7043.00	
mean	0.05	0.23	0.43	
std	0.23	0.42	0.50	
min	0.00	0.00	0.00	
25%	0.00	0.00	0.00	
50%	0.00	0.00	0.00	
75%	0.00	0.00	1.00	
max	1.00	1.00	1.00	

	internet_type_None	offer_Offer A	offer_Offer B	offer_Offer C	\
count	7043.00	7043.00	7043.00	7043.00	
mean	0.22	0.07	0.12	0.06	
std	0.41	0.26	0.32	0.24	
min	0.00	0.00	0.00	0.00	
25%	0.00	0.00	0.00	0.00	
50%	0.00	0.00	0.00	0.00	
75%	0.00	0.00	0.00	0.00	
max	1.00	1.00	1.00	1.00	

	offer_Offer D	offer_Offer E
count	7043.00	7043.00
mean	0.09	0.11
std	0.28	0.32
min	0.00	0.00

```
25%             0.00            0.00
50%             0.00            0.00
75%             0.00            0.00
max             1.00            1.00

[8 rows x 23 columns]
```

1.4 Question 2

- Identify which variables are binary, categorical and not ordinal, categorical and ordinal, and numeric. The non-numeric features will need to be encoded using methods we have discussed in the course.
- Start by identifying the number of unique values each variable takes, then create list variables for categorical, numeric, binary, and ordinal variables.
- Note that the variable 'months' can be treated as numeric, but it may be more convenient to transform it to an ordinal variable.
- For the other categorical variables, examine their values to determine which may be encoded ordinally.

[5]:
```
### BEGIN SOLUTION
df_uniques = pd.DataFrame([[i, len(df[i].unique())] for i in df.columns],␣
 ↪columns=['Variable', 'Unique Values']).set_index('Variable')
df_uniques
```

[5]:

Variable	Unique Values
months	5
multiple	2
gb_mon	50
security	2
backup	2
protection	2
support	2
unlimited	2
contract	3
paperless	2
monthly	1585
satisfaction	5
churn_value	2
payment_Credit Card	2
payment_Mailed Check	2
internet_type_DSL	2
internet_type_Fiber Optic	2
internet_type_None	2
offer_Offer A	2
offer_Offer B	2
offer_Offer C	2

```
offer_Offer D                          2
offer_Offer E                          2
```

[6]:
```
binary_variables = list(df_uniques[df_uniques['Unique Values'] == 2].index)
binary_variables
```

[6]:
```
['multiple',
 'security',
 'backup',
 'protection',
 'support',
 'unlimited',
 'paperless',
 'churn_value',
 'payment_Credit Card',
 'payment_Mailed Check',
 'internet_type_DSL',
 'internet_type_Fiber Optic',
 'internet_type_None',
 'offer_Offer A',
 'offer_Offer B',
 'offer_Offer C',
 'offer_Offer D',
 'offer_Offer E']
```

[7]:
```
categorical_variables = list(df_uniques[(6 >= df_uniques['Unique Values']) &␣
 ↪(df_uniques['Unique Values'] > 2)].index)
categorical_variables
```

[7]:
```
['months', 'contract', 'satisfaction']
```

[8]:
```
[[i, list(df[i].unique())] for i in categorical_variables]
```

[8]:
```
[['months', [0.0, 0.25, 0.5, 1.0, 0.75]],
 ['contract', [0.0, 0.5, 1.0]],
 ['satisfaction', [0.5, 0.25, 0.0, 0.75, 1.0]]]
```

[9]:
```
ordinal_variables = ['contract', 'satisfaction']
```

[10]:
```
df['months'].unique()
```

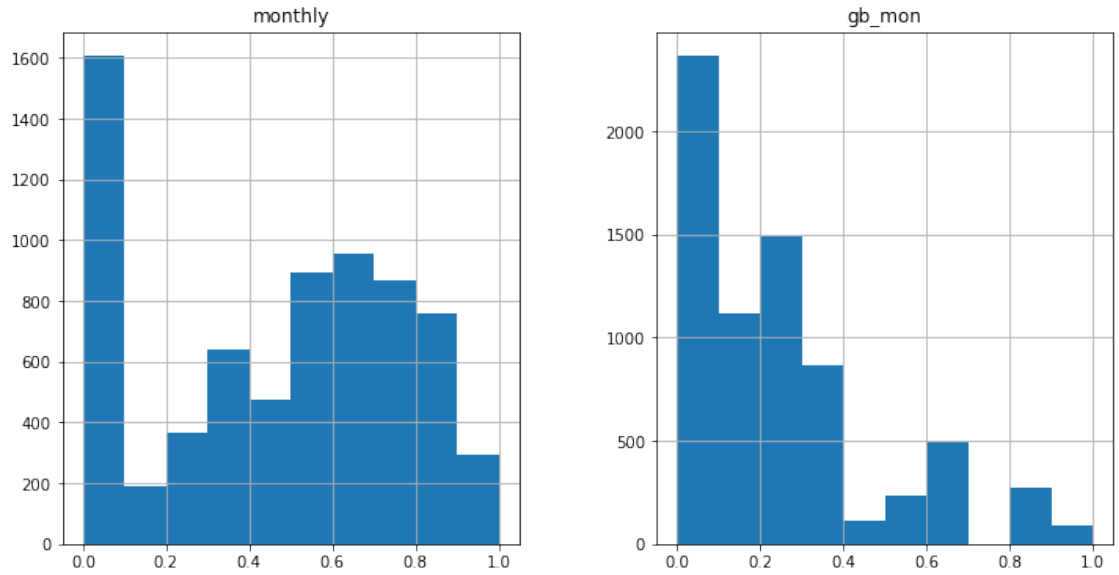
[10]:
```
array([0.  , 0.25, 0.5 , 1.  , 0.75])
```

[11]:
```
ordinal_variables.append('months')
```

[12]:
```
numeric_variables = list(set(df.columns) - set(ordinal_variables) -␣
 ↪set(categorical_variables) - set(binary_variables))
```

[13]:
```
df[numeric_variables].hist(figsize=(12, 6))
```

[13]:
```
array([[<AxesSubplot:title={'center':'monthly'}>,
        <AxesSubplot:title={'center':'gb_mon'}>]], dtype=object)
```



[14]:
```
df['months'] = pd.cut(df['months'], bins=5)
### END SOLUTION
```

1.5 Question 3

- Having set up the variables, remember that the K-nearest neighbors algorithm uses distance and hence requires scaled data.
- Scale the data using one of the scaling methods discussed in the course.
- Save the processed dataframe as a comma-separated file: 'churndata_processed.csv'

[15]:
```
### BEGIN SOLUTION
from sklearn.preprocessing import LabelBinarizer, LabelEncoder, OrdinalEncoder
```

[16]:
```
lb, le = LabelBinarizer(), LabelEncoder()
```

[17]:
```
for column in ordinal_variables:
    df[column] = le.fit_transform(df[column])
```

[18]:
```
df[ordinal_variables].astype('category').describe()
```

[18]:

	contract	satisfaction	months
count	7043	7043	7043
unique	3	5	5

```
top            0             2       0
freq        3610          2665    2470
```

[19]:
```
for column in binary_variables:
    df[column] = lb.fit_transform(df[column])
```

[20]:
```
categorical_variables = list(set(categorical_variables) -␣
 ↪set(ordinal_variables))
```

[21]:
```
df = pd.get_dummies(df, columns = categorical_variables, drop_first=True)
```

[22]:
```
df.describe().T
```

[22]:

	count	mean	std	min	25%	50%	75%	max
months	7043.0	1.734204	1.592924	0.0	0.000000	1.000000	3.000000	4.0
multiple	7043.0	0.421837	0.493888	0.0	0.000000	0.000000	1.000000	1.0
gb_mon	7043.0	0.241358	0.240223	0.0	0.035294	0.200000	0.317647	1.0
security	7043.0	0.286668	0.452237	0.0	0.000000	0.000000	1.000000	1.0
backup	7043.0	0.344881	0.475363	0.0	0.000000	0.000000	1.000000	1.0
protection	7043.0	0.343888	0.475038	0.0	0.000000	0.000000	1.000000	1.0
support	7043.0	0.290217	0.453895	0.0	0.000000	0.000000	1.000000	1.0
unlimited	7043.0	0.673719	0.468885	0.0	0.000000	1.000000	1.000000	1.0
contract	7043.0	0.754792	0.848468	0.0	0.000000			
paperless	7043.0	0.592219	0.491457	0.0	0.000000			
monthly	7043.0	0.462803	0.299403	0.0	0.171642			
satisfaction	7043.0	2.244924	1.201657	0.0	2.000000			
churn_value	7043.0	0.265370	0.441561	0.0	0.000000			
payment_Credit Card	7043.0	0.390317	0.487856	0.0	0.000000			
payment_Mailed Check	7043.0	0.054664	0.227340	0.0	0.000000			
internet_type_DSL	7043.0	0.234559	0.423753	0.0	0.000000			
internet_type_Fiber Optic	7043.0	0.430924	0.495241	0.0	0.000000			
internet_type_None	7043.0	0.216669	0.412004	0.0	0.000000			
offer_Offer A	7043.0	0.073832	0.261516	0.0	0.000000			
offer_Offer B	7043.0	0.116996	0.321438	0.0	0.000000			
offer_Offer C	7043.0	0.058924	0.235499	0.0	0.000000			
offer_Offer D	7043.0	0.085475	0.279607	0.0	0.000000			
offer_Offer E	7043.0	0.114298	0.318195	0.0	0.000000			

```
contract                    0.000000  2.000000  2.0
paperless                   1.000000  1.000000  1.0
monthly                     0.518408  0.712438  1.0
satisfaction                2.000000  3.000000  4.0
churn_value                 0.000000  1.000000  1.0
payment_Credit Card         0.000000  1.000000  1.0
payment_Mailed Check        0.000000  0.000000  1.0
internet_type_DSL           0.000000  0.000000  1.0
internet_type_Fiber Optic   0.000000  1.000000  1.0
internet_type_None          0.000000  0.000000  1.0
offer_Offer A               0.000000  0.000000  1.0
offer_Offer B               0.000000  0.000000  1.0
offer_Offer C               0.000000  0.000000  1.0
offer_Offer D               0.000000  0.000000  1.0
offer_Offer E               0.000000  0.000000  1.0
```

[23]:
```
from sklearn.preprocessing import MinMaxScaler
mm = MinMaxScaler()
```

[24]:
```
for column in [ordinal_variables + numeric_variables]:
    df[column] = mm.fit_transform(df[column])
```

[25]:
```
round(df.describe().T, 3)
```

[25]:

	count	mean	std	min	25%	50%	75%	max
months	7043.0	0.434	0.398	0.0	0.000	0.250	0.750	1.0
multiple	7043.0	0.422	0.494	0.0	0.000	0.000	1.000	1.0
gb_mon	7043.0	0.241	0.240	0.0	0.035	0.200	0.318	1.0
security	7043.0	0.287	0.452	0.0	0.000	0.000	1.000	1.0
backup	7043.0	0.345	0.475	0.0	0.000	0.000	1.000	1.0
protection	7043.0	0.344	0.475	0.0	0.000	0.000	1.000	1.0
support	7043.0	0.290	0.454	0.0	0.000	0.000	1.000	1.0
unlimited	7043.0	0.674	0.469	0.0	0.000	1.000	1.000	1.0
contract	7043.0	0.377	0.424	0.0	0.000	0.000	1.000	1.0
paperless	7043.0	0.592	0.491	0.0	0.000	1.000	1.000	1.0
monthly	7043.0	0.463	0.299	0.0	0.172	0.518	0.712	1.0
satisfaction	7043.0	0.561	0.300	0.0	0.500	0.500	0.750	1.0
churn_value	7043.0	0.265	0.442	0.0	0.000	0.000	1.000	1.0
payment_Credit Card	7043.0	0.390	0.488	0.0	0.000	0.000	1.000	1.0
payment_Mailed Check	7043.0	0.055	0.227	0.0	0.000	0.000	0.000	1.0
internet_type_DSL	7043.0	0.235	0.424	0.0	0.000	0.000	0.000	1.0
internet_type_Fiber Optic	7043.0	0.431	0.495	0.0	0.000	0.000	1.000	1.0
internet_type_None	7043.0	0.217	0.412	0.0	0.000	0.000	0.000	1.0
offer_Offer A	7043.0	0.074	0.262	0.0	0.000	0.000	0.000	1.0
offer_Offer B	7043.0	0.117	0.321	0.0	0.000	0.000	0.000	1.0
offer_Offer C	7043.0	0.059	0.235	0.0	0.000	0.000	0.000	1.0
offer_Offer D	7043.0	0.085	0.280	0.0	0.000	0.000	0.000	1.0

```
offer_Offer E              7043.0  0.114  0.318  0.0  0.000  0.000  0.000  1.0
```

[26]:
```
### END SOLUTION

# Save a copy of the processed data for later use
outputfile = 'churndata_processed.csv'
df.to_csv(outputfile, index=False)
```

1.6 Question 4

- Now that the data are encoded and scaled, separate the features (X) from the target (y, churn_value).
- Split the sample into training and test samples, with the test sample representing 40% of observations.
- Estimate a K-Nearest Neighbors model, using K=3.
- Examine the Precision, Recall, F-1 Score, and Accuracy of the classification.
- Use a graphic to illustrate the Confusion Matrix.

[27]:
```
### BEGIN SOLUTION
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score,␣
 ↪classification_report, f1_score
```

[28]:
```
# Set up X and y variables
y, X = df['churn_value'], df.drop(columns='churn_value')
# Split the data into training and test samples
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,␣
 ↪random_state=42)
```

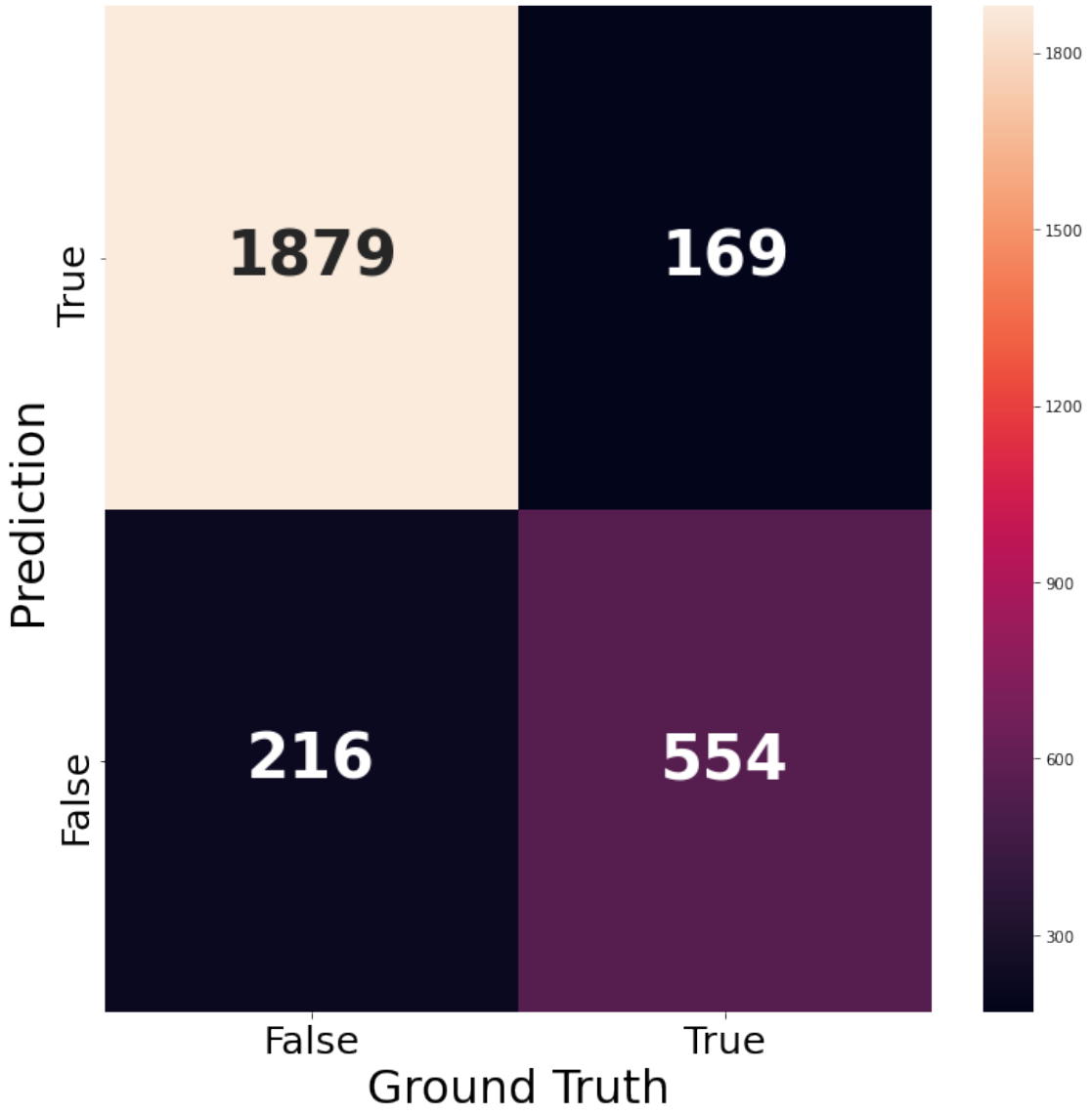
[29]:
```
# Estimate KNN model and report outcomes
knn = KNeighborsClassifier(n_neighbors=3)
knn = knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
# Precision, recall, f-score from the multi-class support function
print(classification_report(y_test, y_pred))
print('Accuracy score: ', round(accuracy_score(y_test, y_pred), 2))
print('F1 Score: ', round(f1_score(y_test, y_pred), 2))
```

```
              precision    recall  f1-score   support

           0       0.90      0.92      0.91      2048
           1       0.77      0.72      0.74       770

   micro avg       0.86      0.86      0.86      2818
   macro avg       0.83      0.82      0.82      2818
weighted avg       0.86      0.86      0.86      2818
```

```
Accuracy score:  0.86
F1 Score:  0.74
```

[30]:
```
# Plot confusion matrix
sns.set_palette(sns.color_palette())
_, ax = plt.subplots(figsize=(12,12))
ax = sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d',␣
 ↪annot_kws={"size": 40, "weight": "bold"})
labels = ['False', 'True']
ax.set_xticklabels(labels, fontsize=25);
ax.set_yticklabels(labels[::-1], fontsize=25);
ax.set_ylabel('Prediction', fontsize=30);
ax.set_xlabel('Ground Truth', fontsize=30)
### END SOLUTION
```

[30]:
```
Text(0.5, 87.0, 'Ground Truth')
```



1.7 Question 5

- Using the same split of training and test samples, estimate another K-Nearest Neighbors model.
- This time, use K=5 and weight the results by distance.
- Again, examine the Precision, Recall, F-1 Score, and Accuracy of the classification, and visualize the Confusion Matrix.

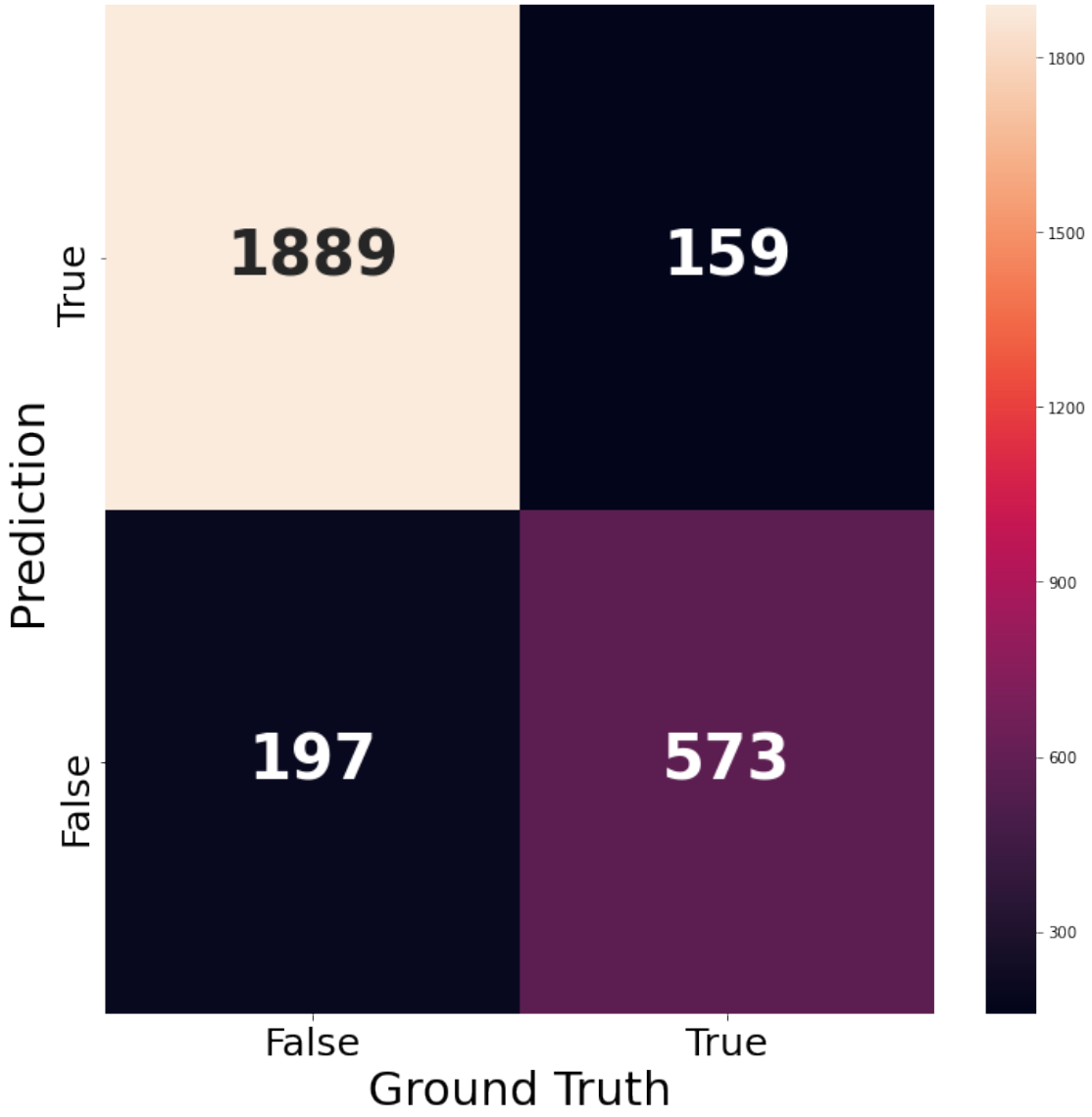
[31]:
```
### BEGIN SOLUTION
knn = KNeighborsClassifier(n_neighbors=5, weights='distance')
knn = knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

```
# Preciision, recall, f-score from the multi-class support function
print(classification_report(y_test, y_pred))
print('Accuracy score: ', round(accuracy_score(y_test, y_pred), 2))
print('F1 Score: ', round(f1_score(y_test, y_pred), 2))
```

```
              precision    recall  f1-score   support

           0       0.91      0.92      0.91      2048
           1       0.78      0.74      0.76       770

   micro avg       0.87      0.87      0.87      2818
   macro avg       0.84      0.83      0.84      2818
weighted avg       0.87      0.87      0.87      2818

Accuracy score:  0.87
F1 Score:  0.76
```

[32]:
```
# Plot confusion matrix
_, ax = plt.subplots(figsize=(12,12))
ax = sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d',␣
 ↪annot_kws={"size": 40, "weight": "bold"})
labels = ['False', 'True']
ax.set_xticklabels(labels, fontsize=25);
ax.set_yticklabels(labels[::-1], fontsize=25);
ax.set_ylabel('Prediction', fontsize=30);
ax.set_xlabel('Ground Truth', fontsize=30)
### END SOLUTION
```

[32]:
```
Text(0.5, 87.0, 'Ground Truth')
```



1.8 Question 6

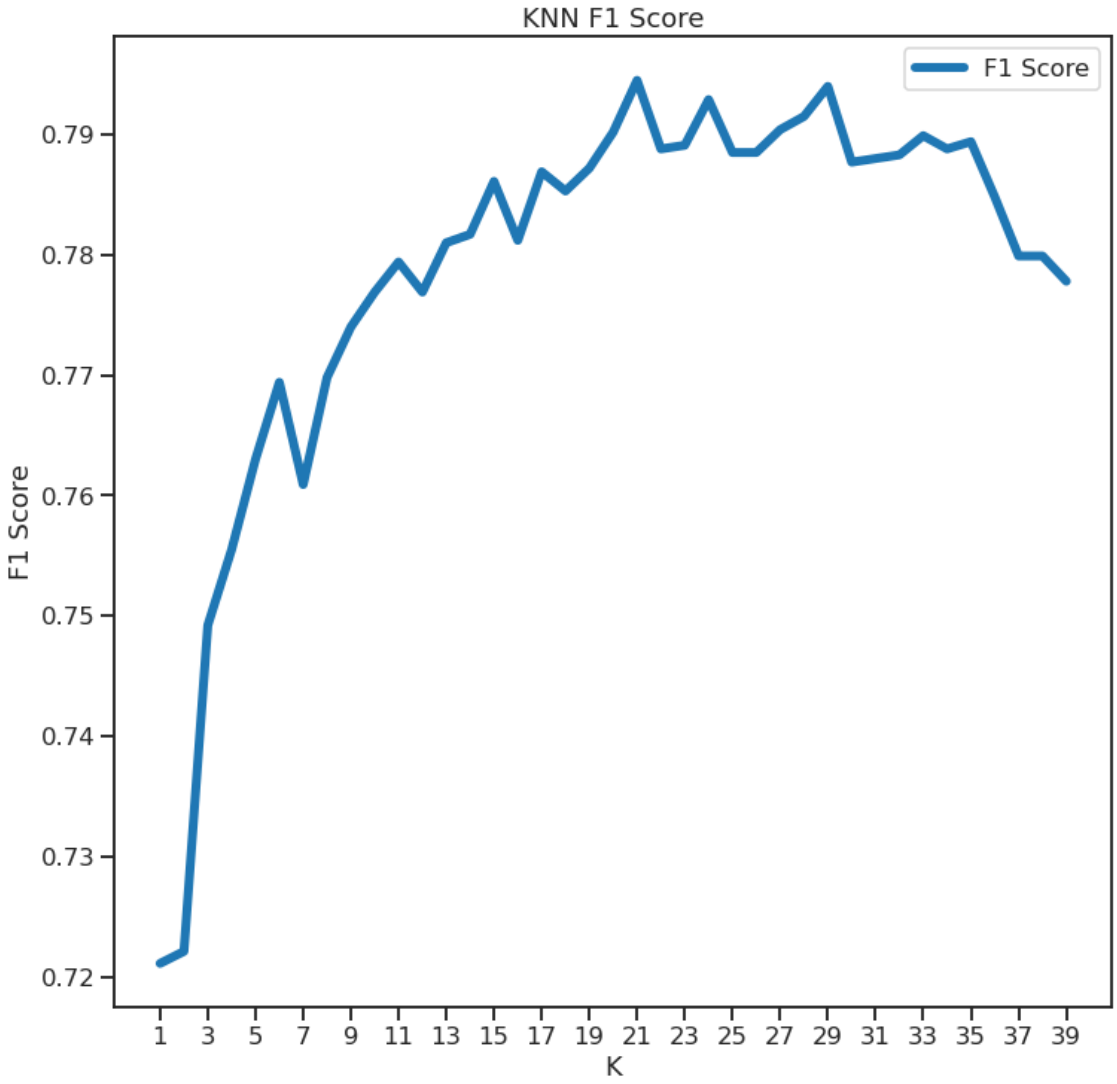
- To determine the right value for K, examine results for values of K from 1 to 40.
- This time, focus on two measures, the F-1 Score, and the Error Rate (1-Accuracy).
- Generate charts which plot each of these measures as a function of K.
- What do these charts suggest about the optimal value for K?

[33]:
```
### BEGIN SOLUTION
max_k = 40
f1_scores = list()
error_rates = list() # 1-accuracy
```

```
for k in range(1, max_k):

    knn = KNeighborsClassifier(n_neighbors=k, weights='distance')
    knn = knn.fit(X_train, y_train)

    y_pred = knn.predict(X_test)
    f1 = f1_score(y_pred, y_test)
    f1_scores.append((k, round(f1_score(y_test, y_pred), 4)))
    error = 1-round(accuracy_score(y_test, y_pred), 4)
    error_rates.append((k, error))

f1_results = pd.DataFrame(f1_scores, columns=['K', 'F1 Score'])
error_results = pd.DataFrame(error_rates, columns=['K', 'Error Rate'])
```

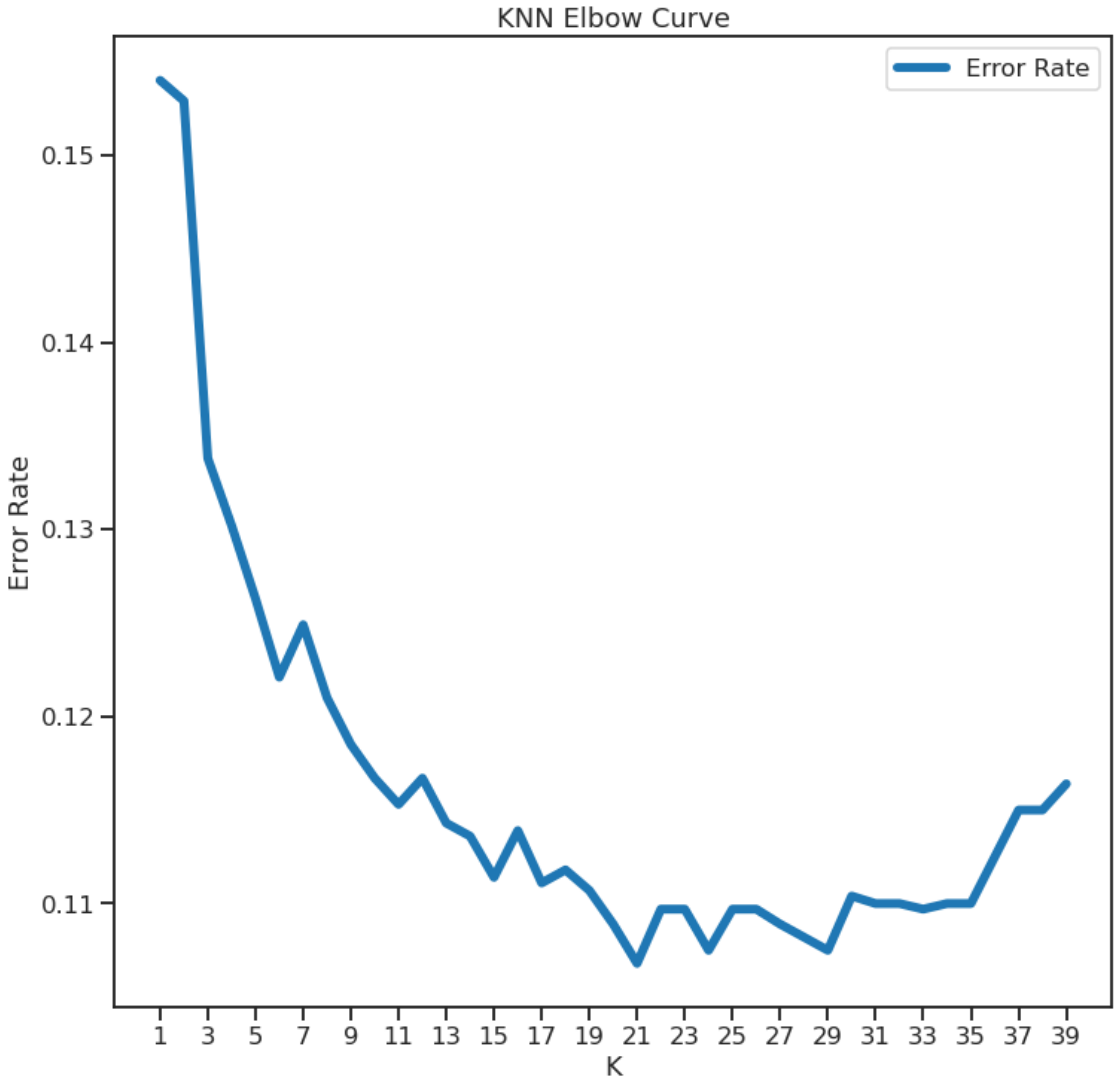
[34]:
```
# Plot F1 results
sns.set_context('talk')
sns.set_style('ticks')

plt.figure(dpi=300)
ax = f1_results.set_index('K').plot(figsize=(12, 12), linewidth=6)
ax.set(xlabel='K', ylabel='F1 Score')
ax.set_xticks(range(1, max_k, 2));
plt.title('KNN F1 Score')
plt.savefig('knn_f1.png')
```

```
<Figure size 1800x1200 with 0 Axes>
```



[35]:
```
# Plot Accuracy (Error Rate) results
sns.set_context('talk')
sns.set_style('ticks')

plt.figure(dpi=300)
ax = error_results.set_index('K').plot(figsize=(12, 12), linewidth=6)
ax.set(xlabel='K', ylabel='Error Rate')
ax.set_xticks(range(1, max_k, 2))
plt.title('KNN Elbow Curve')
plt.savefig('knn_elbow.png')
### END SOLUTION
```

```
<Figure size 1800x1200 with 0 Axes>
```



1.8.1 Machine Learning Foundation (C) 2020 IBM Corporation