rvised-machine-learning-balaram-1

April 18, 2024

1 Week 5 Lab: Supervised Learning

This week's assignment will focus constructing and improving the performance of a KNN model.

1.1 Dataset:

Dataset: bank-additional-full.csv (Provided in folder assign_wk5) ## Assignment Requirements

After reviewing the dataset's descriptive file (bank-additional-names.txt), define a research question to guide your analysis. Make sure your KNN Analysis addresses the following tasks: - Cleanup the dataset as you deem appropriate. As always, defend your reasoning!!! - Missing values? - Column names - Prepare the data for machine learning - A little EDA goeas a long way - Do you need to do anything about data types? - KNN Analysis - What is your optimal K? - Evaluate the accuracy of your model - Discuss ways to improve the performance of your KNN model. * Notice the requirement states **ways** - meaning more than one! * Defend and backup your thoughts!!!!!! - KNN Model Improvement - Implement one of those methods to improve your KNN model performance. - Did your second model perform better than the first? - Conclusion/Summary - Compare and contrast your 2 models
- Include numbers/graphs corresponding to your conclusions - Defend and backup your thoughts!!!!!!

2 Deliverables:

Upload your Jupyter Notebook to the corresponding location in WorldClass.

Note:: Make sure you have clearly indicated each assignment requirement within your notebook.

Important: Make sure you have clearly indicated each assignment requirement within your notebook. Also, I highly encourage you to use markdown text to create a notebook that integrates your analysis within your code. The narrative within your notebook will count for 50% of your total grade on this assignment.

[1]:
```
# Import essential data analysis libraries
import pandas as pd  # For data manipulation and analysis
from sklearn.preprocessing import LabelEncoder  # For converting text labels␣
 ↪into a numeric form

# Visualization libraries to help us see trends and insights
import seaborn as sns  # For making attractive and informative statistical␣
 ↪graphics
```

```python
import matplotlib.pyplot as plt  # For creating static, interactive, and
 ↪animated visualizations in Python

# Machine learning tools from scikit-learn
from sklearn.neighbors import KNeighborsClassifier  # For using the K-Nearest
 ↪Neighbors algorithm
from sklearn.model_selection import train_test_split  # For splitting data into
 ↪training and testing sets
from sklearn.pipeline import Pipeline  # For sequential application of a list
 ↪of transforms and a final estimator
from sklearn.preprocessing import StandardScaler  # For scaling features to
 ↪standardize the range of independent variables
```

Problem statement : Identify key factors that predict if a client will subscribe to a bank's term deposit program.

Load and veiw the data file

[2]:
```python
import pandas as pd

# Load the dataset
df = pd.read_csv('C:/Users/balar/Downloads/assign_wk5/assign_wk5/
 ↪bank-additional-full.csv')

# Display the first few rows of the dataframe
df.head(20)
```

[2]:
```
   age;"job";"marital";"education";"default";"housing";"loan";"contact";"month";
"day_of_week";"duration";"campaign";"pdays";"previous";"poutcome";"emp.var.rate"
;"cons.price.idx";"cons.conf.idx";"euribor3m";"nr.employed";"y"
0   56;"housemaid";"married";"basic.4y";"no";"no";…
1   57;"services";"married";"high.school";"unknown…
2   37;"services";"married";"high.school";"no";"ye…
3   40;"admin.";"married";"basic.6y";"no";"no";"no…
4   56;"services";"married";"high.school";"no";"no…
5   45;"services";"married";"basic.9y";"unknown";"…
6   59;"admin.";"married";"professional.course";"n…
7   41;"blue-collar";"married";"unknown";"unknown"…
8   24;"technician";"single";"professional.course"…
9   25;"services";"single";"high.school";"no";"yes…
10  41;"blue-collar";"married";"unknown";"unknown"…
11  25;"services";"single";"high.school";"no";"yes…
12  29;"blue-collar";"single";"high.school";"no";"…
13  57;"housemaid";"divorced";"basic.4y";"no";"yes…
14  35;"blue-collar";"married";"basic.6y";"no";"ye…
15  54;"retired";"married";"basic.9y";"unknown";"y…
16  35;"blue-collar";"married";"basic.6y";"no";"ye…
```

```
17  46;"blue-collar";"married";"basic.6y";"unknown…
18  50;"blue-collar";"married";"basic.9y";"no";"ye…
19  39;"management";"single";"basic.9y";"unknown";…
```

Use separator and veiw data frame

[3]:
```
df = pd.read_csv('C:/Users/balar/Downloads/assign_wk5/assign_wk5/
↪bank-additional-full.csv', sep=';')
df
```

[3]:
```
       age          job  marital            education  default housing loan  \
0       56    housemaid  married             basic.4y       no      no   no
1       57     services  married          high.school  unknown      no   no
2       37     services  married          high.school       no     yes   no
3       40       admin.  married             basic.6y       no      no   no
4       56     services  married          high.school       no      no  yes
…      …          …        …                  …       …     …    …
41183   73      retired  married  professional.course       no     yes   no
41184   46  blue-collar  married  professional.course       no      no   no
41185   56      retired  married    university.degree       no     yes   no
41186   44   technician  married  professional.course       no      no   no
41187   74      retired  married  professional.course       no     yes   no

         contact month day_of_week  …  campaign  pdays  previous  \
0      telephone   may         mon  …         1    999         0
1      telephone   may         mon  …         1    999         0
2      telephone   may         mon  …         1    999         0
3      telephone   may         mon  …         1    999         0
4      telephone   may         mon  …         1    999         0
…          …   …           …  …       …    …       …
41183   cellular   nov         fri  …         1    999         0
41184   cellular   nov         fri  …         1    999         0
41185   cellular   nov         fri  …         2    999         0
41186   cellular   nov         fri  …         1    999         0
41187   cellular   nov         fri  …         3    999         1

          poutcome emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  \
0      nonexistent          1.1          93.994          -36.4      4.857
1      nonexistent          1.1          93.994          -36.4      4.857
2      nonexistent          1.1          93.994          -36.4      4.857
3      nonexistent          1.1          93.994          -36.4      4.857
4      nonexistent          1.1          93.994          -36.4      4.857
…              …          …             …            …        …
41183  nonexistent         -1.1          94.767          -50.8      1.028
41184  nonexistent         -1.1          94.767          -50.8      1.028
41185  nonexistent         -1.1          94.767          -50.8      1.028
41186  nonexistent         -1.1          94.767          -50.8      1.028
```

```
41187      failure        -1.1          94.767          -50.8      1.028

       nr.employed    y
0           5191.0   no
1           5191.0   no
2           5191.0   no
3           5191.0   no
4           5191.0   no
…              …  …
41183       4963.6  yes
41184       4963.6   no
41185       4963.6   no
41186       4963.6  yes
41187       4963.6   no

[41188 rows x 21 columns]
```

Check shape, info, describe for our data frame

[4]:
```
df.shape
```

[4]:
```
(41188, 21)
```

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41188 non-null  int64
 1   job             41188 non-null  object
 2   marital         41188 non-null  object
 3   education       41188 non-null  object
 4   default         41188 non-null  object
 5   housing         41188 non-null  object
 6   loan            41188 non-null  object
 7   contact         41188 non-null  object
 8   month           41188 non-null  object
 9   day_of_week     41188 non-null  object
 10  duration        41188 non-null  int64
 11  campaign        41188 non-null  int64
 12  pdays           41188 non-null  int64
 13  previous        41188 non-null  int64
 14  poutcome        41188 non-null  object
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
```

```
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  object
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
```

The dataset doesn't have any blank spots, as we've seen from checking all the rows and columns with the `info()` method. However, the description tells us that missing information is actually labeled as 'unknown'. We should take a closer look at the data before we start cleaning it up.

[6]:
```
df.describe()
```

[6]:

	age	duration	campaign	pdays	previous
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000
mean	40.02406	258.285010	2.567593	962.475454	0.172963
std	10.42125	259.279249	2.770014	186.910907	0.494901
min	17.00000	0.000000	1.000000	0.000000	0.000000
25%	32.00000	102.000000	1.000000	999.000000	0.000000
50%	38.00000	180.000000	2.000000	999.000000	0.000000
75%	47.00000	319.000000	3.000000	999.000000	0.000000
max	98.00000	4918.000000	56.000000	999.000000	7.000000

	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
count	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000
mean	0.081886	93.575664	-40.502600	3.621291	5167.035911
std	1.570960	0.578840	4.628198	1.734447	72.251528
min	-3.400000	92.201000	-50.800000	0.634000	4963.600000
25%	-1.800000	93.075000	-42.700000	1.344000	5099.100000
50%	1.100000	93.749000	-41.800000	4.857000	5191.000000
75%	1.400000	93.994000	-36.400000	4.961000	5228.100000
max	1.400000	94.767000	-26.900000	5.045000	5228.100000

Observations : The typical client is around 40 years old.
Calls usually take around 4 to 5 minutes, but some can last for hours.
Clients are usually called 2 or 3 times during a campaign.

As per our problem statement (Identify key factors that predict if a client will subscribe to a bank's term deposit program.) and by observing data we taking demographic and loan data coloumns which are the factors for bank deposit programs and not considering other coloumns

[47]:
```
# taking coloumns which are considered as factors
new_df = df[['age', 'job', 'marital', 'education', 'default', 'housing',␣
↪'loan', 'y']]
new_df
```

[47]:

	age	job	marital	education	default	housing	loan	\
0	56	housemaid	married	basic.4y	no	no	no	
1	57	services	married	high.school	unknown	no	no	
2	37	services	married	high.school	no	yes	no	

```
3        40       admin.  married            basic.6y       no       no   no
4        56     services  married         high.school       no       no  yes
…     …          …       …                      …       …      …   …
41183    73      retired  married  professional.course       no      yes   no
41184    46  blue-collar  married  professional.course       no       no   no
41185    56      retired  married    university.degree       no      yes   no
41186    44   technician  married  professional.course       no       no   no
41187    74      retired  married  professional.course       no      yes   no

          y
0        no
1        no
2        no
3        no
4        no
…     …
41183   yes
41184    no
41185    no
41186   yes
41187    no

[41188 rows x 8 columns]
```

Check the info and then Outliers

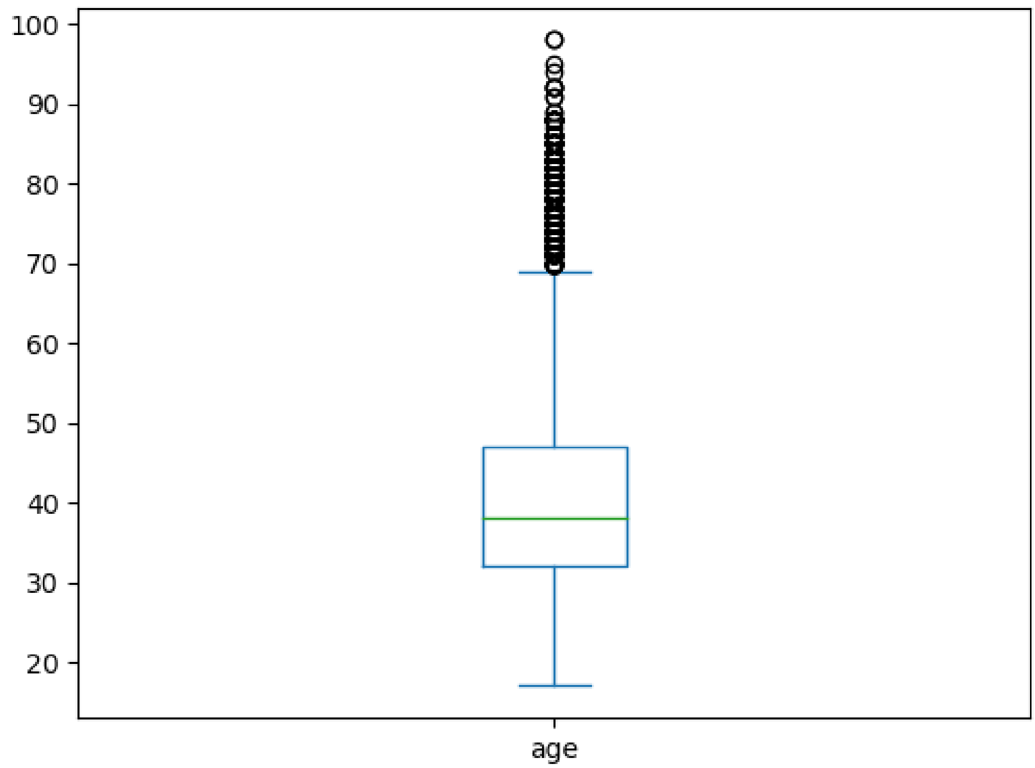
[8]:
```
new_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 8 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        41188 non-null  int64
 1   job        41188 non-null  object
 2   marital    41188 non-null  object
 3   education  41188 non-null  object
 4   default    41188 non-null  object
 5   housing    41188 non-null  object
 6   loan       41188 non-null  object
 7   y          41188 non-null  object
dtypes: int64(1), object(7)
memory usage: 2.5+ MB
```

Cleaning of Data

[10]:
```
new_df.age.plot(kind='box')
```

[10]: <Axes: >



We can see many outliers

Re modify our data frame with consideration of all loans

[12]:
```
new_df = new_df.rename(columns=
                            {'y':'subscription_outcome',
                             'marital':'marital_status',
                             'default':'previously_defaulted_loan',
                             'housing':'has_housing_loan',
                             'loan':'has_personal_loan'})

new_df
```

[12]:
```
         age         job marital_status           education  \
0         56   housemaid        married            basic.4y
1         57    services        married         high.school
2         37    services        married         high.school
3         40      admin.        married            basic.6y
4         56    services        married         high.school
…       …         …              …                  …
41183   73     retired        married  professional.course
```

```
41184   46  blue-collar        married  professional.course
41185   56      retired        married    university.degree
41186   44   technician        married  professional.course
41187   74      retired        married  professional.course

      previously_defaulted_loan has_housing_loan has_personal_loan  \
0                            no               no                no
1                       unknown               no                no
2                            no              yes                no
3                            no               no                no
4                            no               no               yes
…                           …              …                …
41183                        no              yes                no
41184                        no               no                no
41185                        no              yes                no
41186                        no               no                no
41187                        no              yes                no

      subscription_outcome
0                       no
1                       no
2                       no
3                       no
4                       no
…                    …
41183                  yes
41184                   no
41185                   no
41186                  yes
41187                   no

[41188 rows x 8 columns]
```

[13]:
```
new_df.age.describe()
```

[13]:
```
count    41188.00000
mean        40.02406
std         10.42125
min         17.00000
25%         32.00000
50%         38.00000
75%         47.00000
max         98.00000
Name: age, dtype: float64
```

Check value counts

[14]:
```
new_df.job.value_counts()
```

[14]:
```
job
admin.           10422
blue-collar       9254
technician        6743
services          3969
management        2924
retired           1720
entrepreneur      1456
self-employed     1421
housemaid         1060
unemployed        1014
student            875
unknown            330
Name: count, dtype: int64
```

[15]:
```
new_df.marital_status.value_counts()
```

[15]:
```
marital_status
married     24928
single      11568
divorced     4612
unknown        80
Name: count, dtype: int64
```

[16]:
```
new_df.education.value_counts()
```

[16]:
```
education
university.degree      12168
high.school             9515
basic.9y                6045
professional.course     5243
basic.4y                4176
basic.6y                2292
unknown                 1731
illiterate                18
Name: count, dtype: int64
```

[17]:
```
new_df.previously_defaulted_loan.value_counts()
```

[17]:
```
previously_defaulted_loan
no         32588
unknown     8597
yes            3
Name: count, dtype: int64
```

[18]:
```
new_df.has_housing_loan.value_counts()
```

[18]:
```
has_housing_loan
yes        21576
no         18622
unknown      990
Name: count, dtype: int64
```

[19]:
```
new_df.has_personal_loan.value_counts()
```

[19]:
```
has_personal_loan
no         33950
yes         6248
unknown      990
Name: count, dtype: int64
```

for converting caterogical to numerical use label encoding

[21]:
```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
```

[22]:
```
new_df['job'] = label_encoder.fit_transform(new_df['job'])
new_df.marital_status = label_encoder.fit_transform(new_df.marital_status)
new_df.education = label_encoder.fit_transform(new_df.education)
new_df.previously_defaulted_loan = label_encoder.fit_transform(new_df.
  ↪previously_defaulted_loan)
new_df.has_housing_loan = label_encoder.fit_transform(new_df.has_housing_loan)
new_df.has_personal_loan = label_encoder.fit_transform(new_df.has_personal_loan)
new_df.subscription_outcome = label_encoder.fit_transform(new_df.
  ↪subscription_outcome)
```

[23]:
```
new_df
```

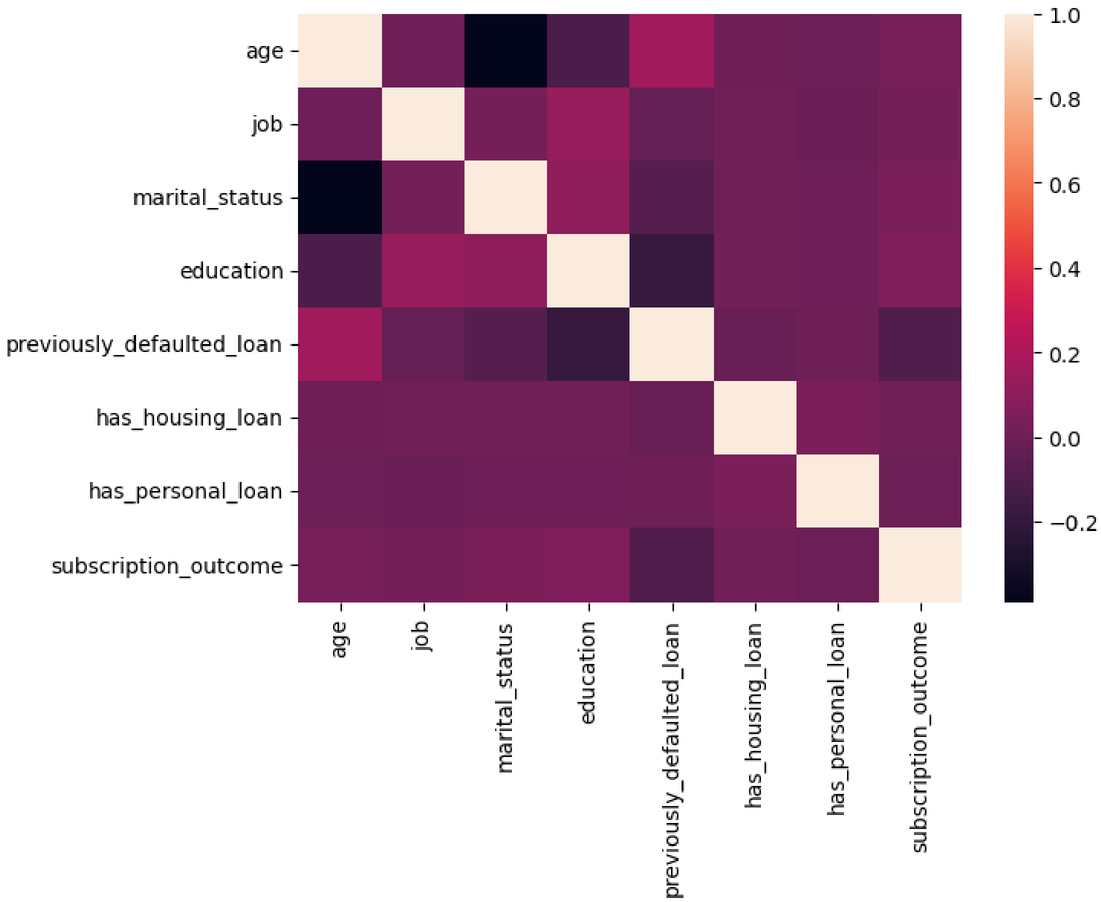
[23]:
```
       age  job  marital_status  education  previously_defaulted_loan  \
0       56    3               1          0                          0
1       57    7               1          3                          1
2       37    7               1          3                          0
3       40    0               1          1                          0
4       56    7               1          3                          0
…      …    …             …          …                          …
41183   73    5               1          5                          0
41184   46    1               1          5                          0
41185   56    5               1          6                          0
41186   44    9               1          5                          0
41187   74    5               1          5                          0

       has_housing_loan  has_personal_loan  subscription_outcome
```

0	0	0	0	
1	0	0	0	
2	2	0	0	
3	0	0	0	
4	0	2	0	
…	…	…	…	
41183	2	0	1	
41184	0	0	0	
41185	2	0	0	
41186	0	0	1	
41187	2	0	0	

[41188 rows x 8 columns]

Plot a heat map

[29]:
```
plt.figure(figsize=(7,5))
_ = sns.heatmap(new_df.corr())
```



There are some relationships between two features eg age and previously_defaulted_loan, education and job etc, but the relationships are not very strong (you can clearly observe from heat map)

For creating Model, Select features and Split the dataset into training and test sets

[36]:
```
# Assume 'new_df' is your DataFrame and the last column 'y' is the target␣
↪variable

# Select features (all columns except the last one) and the target (the last␣
↪column)
X = new_df.iloc[:, :-1]
y = new_df.iloc[:, -1]

# Split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[37]:
```
model = KNeighborsClassifier(n_neighbors=4, n_jobs=-1)
model.fit(X_train, y_train)
score = model.score(X_test, y_test)
score
```

[37]:
```
0.8816460305899491
```

we got a pretty good model score, This means that even though the details we have don't seem to relate to each other very much, we can still combine them to make a model that can guess if a client might subscribe for loan

Check accuracy (i.e Optimal value of K)with different number of neighbours

[38]:
```
accuracy_score = []

for k in range(1, 11):
    model = KNeighborsClassifier(n_neighbors=k, n_jobs=-1)
    model.fit(X_train, y_train)
    score = model.score(X_test, y_test)
    accuracy_score.append(score)

    print(f'K value {k}: {score}')
```

```
K value 1: 0.8159747511531925
K value 2: 0.8771546491866958
K value 3: 0.8622238407380433
K value 4: 0.8816460305899491
K value 5: 0.872784656470017
K value 6: 0.8815246419033747
K value 7: 0.8792182568584608
K value 8: 0.8839524156348628
```

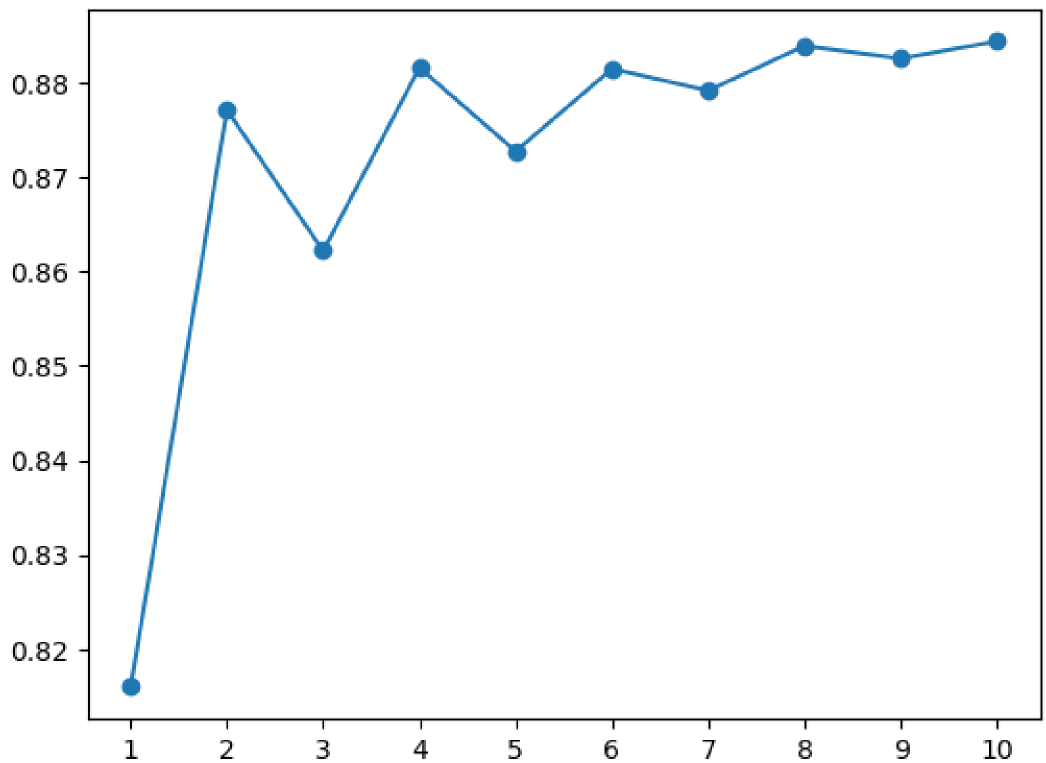
```
K value 9: 0.8826171400825443
K value 10: 0.8844379703811605
```

[39]:
```
max(accuracy_score)
```

[39]:
```
0.8844379703811605
```

Max accuracy is observed at k=4 and see the plot for all varying of K

[42]:
```
plt.scatter(range(1, 11), accuracy_score);
plt.plot(range(1, 11), accuracy_score);
plt.xticks(range(1, 11));
```



Making Your KNN Model Better To make our KNN model better, we can:

1. Adjust settings like how many neighbors the model should consider. Changing these can make the model predict better, but just adding more neighbors doesn't always help.
2. Cut down on less important data but keep the crucial stuff. Our data is already pretty trimmed, so cutting more might lose important bits.
3. Make sure all the data is on a similar scale so that no single thing overpowers the rest. This can help since things like age and job are on different scales.

We'll try scaling our data to see if our model gets better.

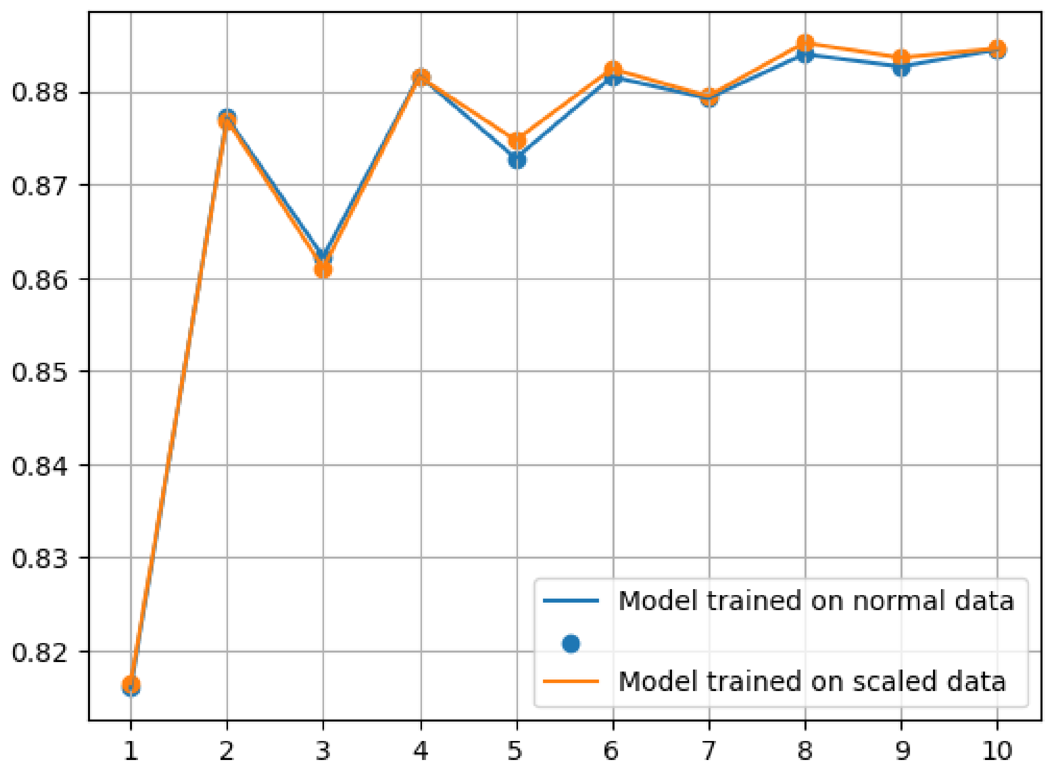
[ ]:
```
Use Standardscalar and compare results with KNN accuracies
```

[43]:
```
norm_trained_model_scores = []
scaled_dataTrained_model = []
for k in range(1, 11):
    data_scaled_model = Pipeline([('Scaler', StandardScaler()),('KNN',␣
 ↪KNeighborsClassifier(n_neighbors=k, n_jobs=-1))])
    data_scaled_model.fit(X_train, y_train)
    data_scaled_model_score = data_scaled_model.score(X_test, y_test)
    scaled_dataTrained_model += [data_scaled_model_score]



    norm_trained_model = KNeighborsClassifier(n_neighbors=k, n_jobs=-1)
    norm_trained_model.fit(X_train, y_train)
    norm_trained_model_score = norm_trained_model.score(X_test, y_test)
    norm_trained_model_scores += [norm_trained_model_score]


    print(f'K value {k}: scaled data model accuracy score:␣
 ↪{data_scaled_model_score} | Normal data mode accuracu score:␣
 ↪{norm_trained_model_score}')
```

```
K value 1: scaled data model accuracy score: 0.8164603058994901 | Normal data
mode accuracu score: 0.8159747511531925
K value 2: scaled data model accuracy score: 0.876911871813547 | Normal data
mode accuracu score: 0.8771546491866958
K value 3: scaled data model accuracy score: 0.8610099538722991 | Normal data
mode accuracu score: 0.8622238407380433
K value 4: scaled data model accuracy score: 0.8815246419033747 | Normal data
mode accuracu score: 0.8816460305899491
K value 5: scaled data model accuracy score: 0.8747268754552076 | Normal data
mode accuracu score: 0.872784656470017
K value 6: scaled data model accuracy score: 0.8823743627093955 | Normal data
mode accuracu score: 0.8815246419033747
K value 7: scaled data model accuracy score: 0.8794610342316096 | Normal data
mode accuracu score: 0.8792182568584608
K value 8: scaled data model accuracy score: 0.8851663025006069 | Normal data
mode accuracu score: 0.8839524156348628
K value 9: scaled data model accuracy score: 0.8835882495751396 | Normal data
mode accuracu score: 0.8826171400825443
K value 10: scaled data model accuracy score: 0.8845593590677349 | Normal data
mode accuracu score: 0.8844379703811605
```

[45]:
```python
# display the resutls
plt.plot(range(1, 11), norm_trained_model_scores)
plt.scatter(range(1, 11), norm_trained_model_scores)

plt.plot(range(1, 11), scaled_dataTrained_model)
plt.scatter(range(1, 11), scaled_dataTrained_model)

plt.grid()
plt.xticks(range(1, 11))
plt.legend(['Model trained on normal data', None, 'Model trained on scaled␣
↪data'])
```

[45]: <matplotlib.legend.Legend at 0x2203d361dd0>



Looking at how well the two models did, we can tell they're both doing well.they're at least 80% accurate. The best models are more than 88.44% accurate.

conclusion Our model shows key factors that predict if a client will subscribe to a bank's term deposit program. just by using basic information about them and their loan history. It's impressive that our model reached an accuracy of 88.44% with this limited information.

To make the model even better, we need to scale the data. This means adjusting the features so they're all measured the same way. Doing this helps prevent any single feature from having too much influence and can make our predictions more accurate.

```
[ ]:
```

```
[ ]:
```