HW 4

This assignment covers Linear Classification methods

DO NOT ERASE MARKDOWN CELLS AND INSTRUCTIONS IN YOUR HW submission

- **Q** - QUESTION
- **A** - Where to input your answer

Instructions

Keep the following in mind for all notebooks you develop:

- Structure your notebook.
- Use headings with meaningful levels in Markdown cells, and explain the questions each piece of code is to answer or the reason it is there.
- Make sure your notebook can always be rerun from top to bottom.
- Please start working on this assignment as soon as possible. If you are a beginner in Python this might take a long time. One of the objectives of this assignment is to help you learn python and scikit-learn package.
- See README.md for homework submission instructions

Related Tutorials

Refreshers

- Intro to Machine Learning w scikit-learn
- A tutorial on statistical-learning for scientific data processing

Classification Approaches

- Logistic Regression with Sklearn
- KNN with sklearn

Modeling

- Cross-validation
- Plot Confursion Matrix with Sklearn
- Confusion Matrix Display

Data Processing

Q1 Get training data from the dataframe

1. Load mobile_data.csv from ```data'' folder into the dataframe
2. Assign values of `price_range` column to `y`
3. Drop 'price_range' column from data frame,
4. Assign remaining df column values to x
5. Print the head of the dataframe

A1 Replace ??? with code in the code cell below

In [ ]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from matplotlib import pyplot as plt
from sklearn.preprocessing import StandardScaler

#Read the mobile_data.csv file using the prropriate separator as input to read_csv()
df = pd.read_csv('data/mobile_data.csv')

df.head()
```

Out[ ]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores
0	842	0	2.2	0	1	0	7	0.6	188	2
1	1021	1	0.5	1	0	1	53	0.7	136	3
2	563	1	0.5	1	2	1	41	0.9	145	5
3	615	1	2.5	0	0	0	10	0.8	131	6
4	1821	1	1.2	0	13	1	44	0.6	141	2

5 rows × 21 columns

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

Out[ ]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	20
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.046500	
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.145715	
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000	
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.000000	
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.000000	
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000	
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000	

8 rows × 21 columns

In [ ]:
```
y = df['price_range']

# concerning part 3, I decided to combine it with part 4. I want to keep the original
x = df.drop(columns=['price_range'])
```

In [ ]:
```
df.head()
```

Out[ ]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores
0	842	0	2.2	0	1	0	7	0.6	188	2
1	1021	1	0.5	1	0	1	53	0.7	136	3
2	563	1	0.5	1	2	1	41	0.9	145	5
3	615	1	2.5	0	0	0	10	0.8	131	6
4	1821	1	1.2	0	13	1	44	0.6	141	2

5 rows × 21 columns

Q2:

1. Check number of null values per column in the x dataframe.

A2 Replace ??? with code in the code cell below

In [ ]:
```
# sum() will tally each column's empty cell. Value will be > 0 if there are empty cell
x.isnull().sum()
```

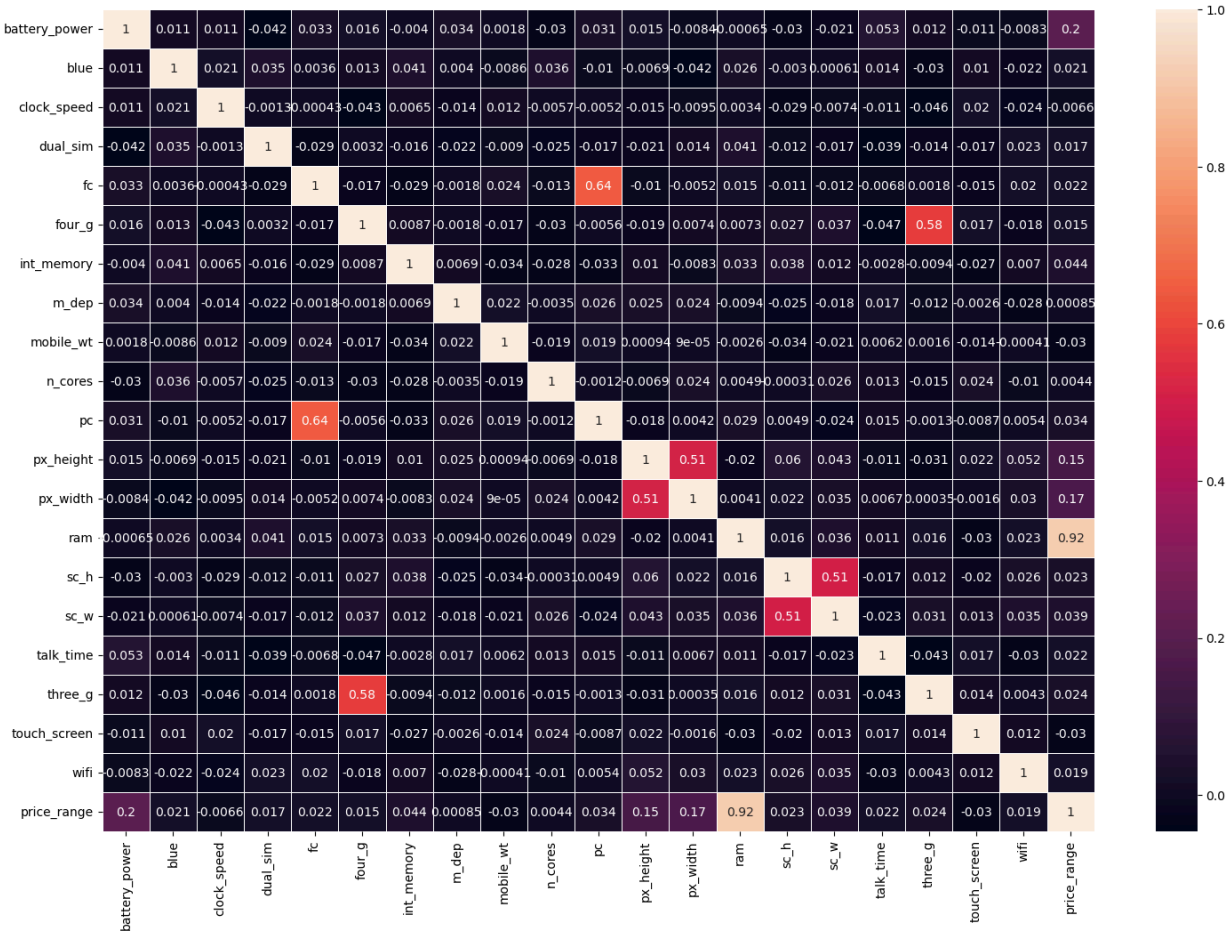
Out[ ]:
```
battery_power    0
blue             0
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w             0
talk_time        0
three_g          0
touch_screen     0
wifi             0
dtype: int64
```

Q3.1 Use seaborn heatmap chart to visualize the correlations between the columns. Replace ??? with code in the code cell below

A3.1

In [ ]:
```
import seaborn as sns
plt.figure(figsize=(18,12))
sns.heatmap(data=df.corr(), linecolor='white', linewidths=0.5, annot=True)
```

Out[ ]:
```
<Axes: >
```



Q3.2 List columns that correlate the most with the 'price_range' column.

Note: For this dataset any column that has correlation factor over or near 0.1 can be considered as a good predictor/ feature.

A3.2 ram, px_width, px_height, battery_power

Q3.3 Update the 'x' dataframe defined earlier in Q1 with your selected features/columns for 'price_range'.

A3.3 Replace ??? with code in the code cell below

In [ ]:
```
# A3 Part 3:

x = df[['ram', 'px_width', 'px_height', 'battery_power']]
x.head(10)
```

Out[ ]:

	ram	px_width	px_height	battery_power
0	2549	756	20	842
1	2631	1988	905	1021
2	2603	1716	1263	563
3	2769	1786	1216	615
4	1411	1212	1208	1821
5	1067	1654	1004	1859
6	3220	1018	381	1821
7	700	1149	512	1954
8	1099	836	386	1445
9	513	1224	1137	509

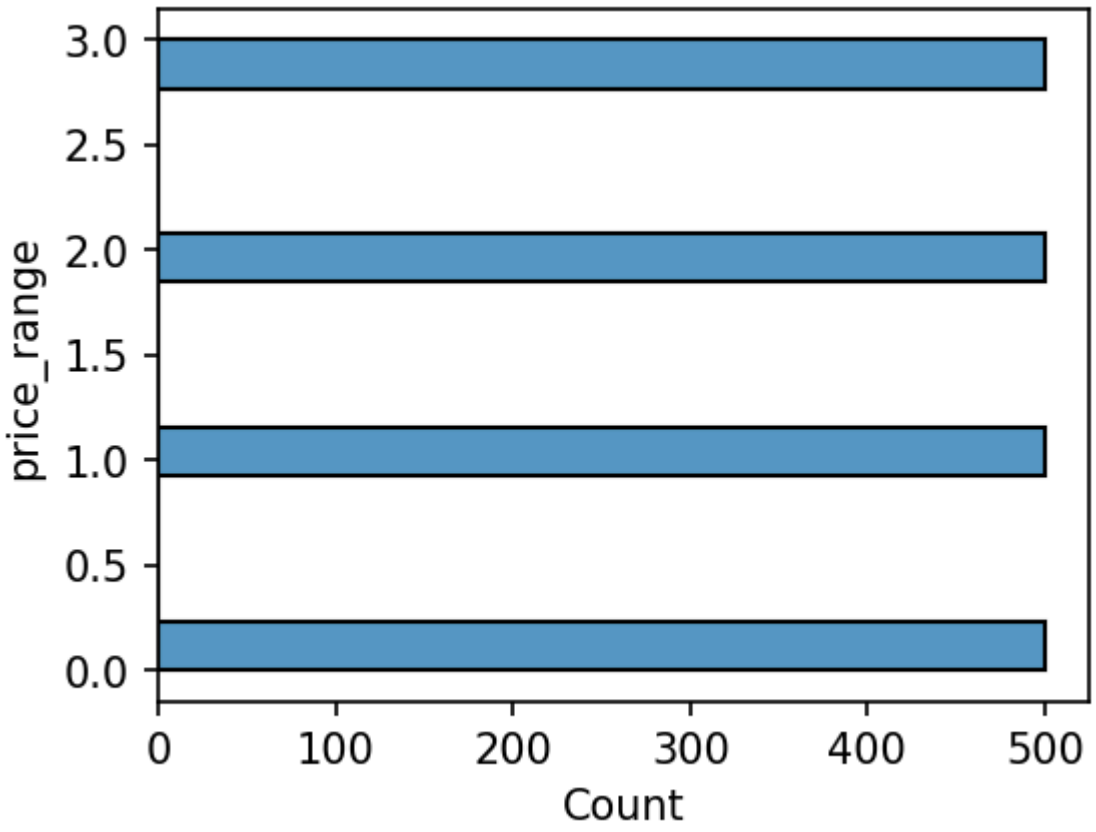
Q4: Use seaborn *histplot* to plot a distribution graph for the price_range column

A4 Replace ??? with code in the code cell below

In [ ]:
```
plt.figure(figsize=(4,3),dpi=150)
#sns.?
sns.histplot(data=x, y=y)
```

```
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[ ]:
```
<Axes: xlabel='Count', ylabel='price_range'>
```



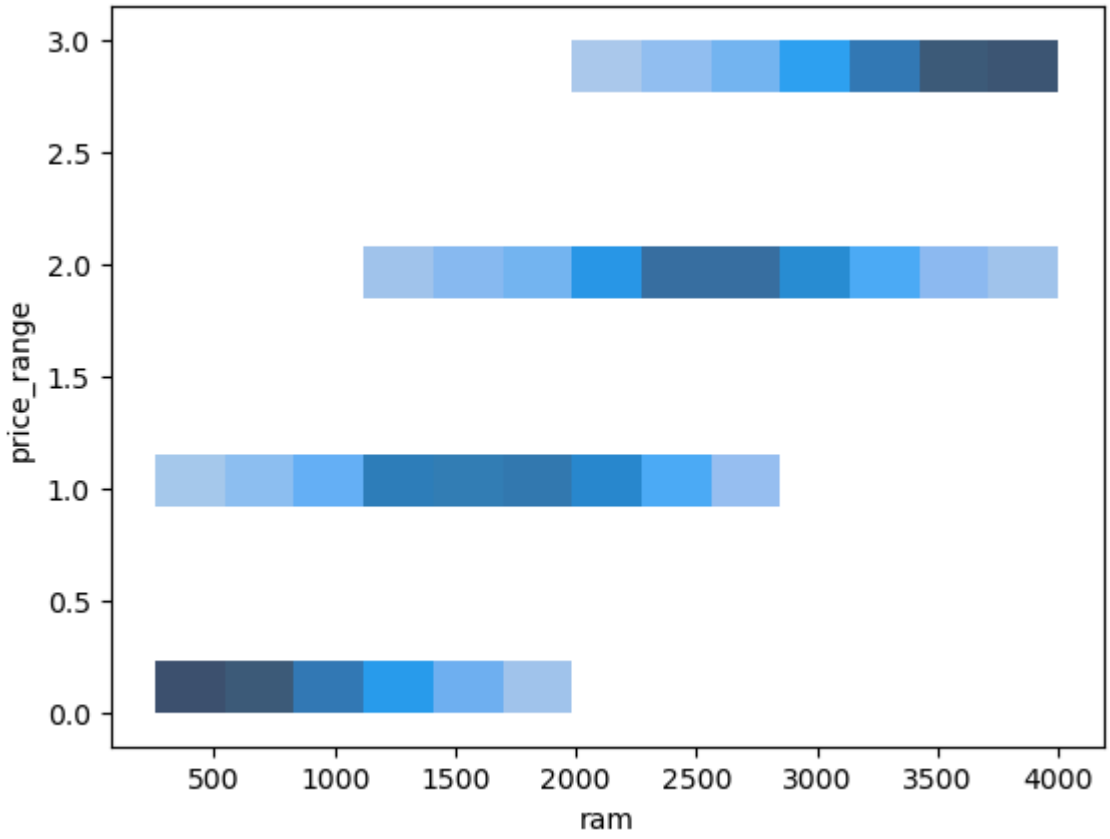
Q5: Use seaborn *histplot* to present the relation between _price*range* and the *ram* of a mobile

A5 Replace ??? with code in the code cell below

In [ ]:
```
#sns.?
sns.histplot(x=x['ram'], y=y)
```

```
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[ ]:
```
<Axes: xlabel='ram', ylabel='price_range'>
```



Q6:

1. Use StandardScaler from sklearn to transform the x dataframe.
2. Split dataset into train and test data use train_test_split with test_size = 0.2 and random_state = 42
3. Check the number of instance in the train and test set.
4. Check the number of instance per class in train and test set using ytrain and ytest

A6 Replace ??? with code in the code cell below

In [ ]:
```
#scaler = ?
#scaler.?
#x= scaler.?

scaler = StandardScaler()
scaler.fit(x)
x = scaler.transform(x)
#print(x)
```

In [ ]:
```
xtrain, xtest, ytrain, ytest = train_test_split(x, y, random_state=42, test_size=0.2)
```

In [ ]:
```
#print(?)
#print(?)
print(xtest)
print(ytest)
```

```
[[-1.32620097  0.82502108 -0.97844827  0.92755239]
 [ 0.40461305 -0.61448923 -0.83419687 -0.12865337]
 [-0.74526712 -0.69317628 -0.79362616  1.669628  ]
 ...
 [-0.30449509  0.02657887  1.31830459 -0.44278354]
 [-0.07857637  0.64218939 -0.26846088 -1.16209609]
 [ 0.29764745  0.90370814  0.43251079 -0.12182446]]
1860    0
353     2
1333    1
905     3
1289    1
       ..
965     3
1284    2
1739    1
261     1
535     2
Name: price_range, Length: 400, dtype: int64
```

In [ ]:
```
#ytrain.?
ytrain.shape
```

Out[ ]:
```
(1600,)
```

In [ ]:
```
#ytest.?
ytest.shape
```

Out[ ]:
```
(400,)
```

Classification Model 1: Logistic Regression

Here, we fit Logistic Regression model to the train dataset using K-fold cross validation

Q7 Train Logistic Regression Model

1. Create a logistic regression model using sklearn linear_model library.
2. Fit the model with the train data
3. Get the score from the model using test data
4. Plot confusion matrix using ConfusionMatrixDisplay, see Visualization with Display Objects example.

A7 Replace ??? with code in the code cell below

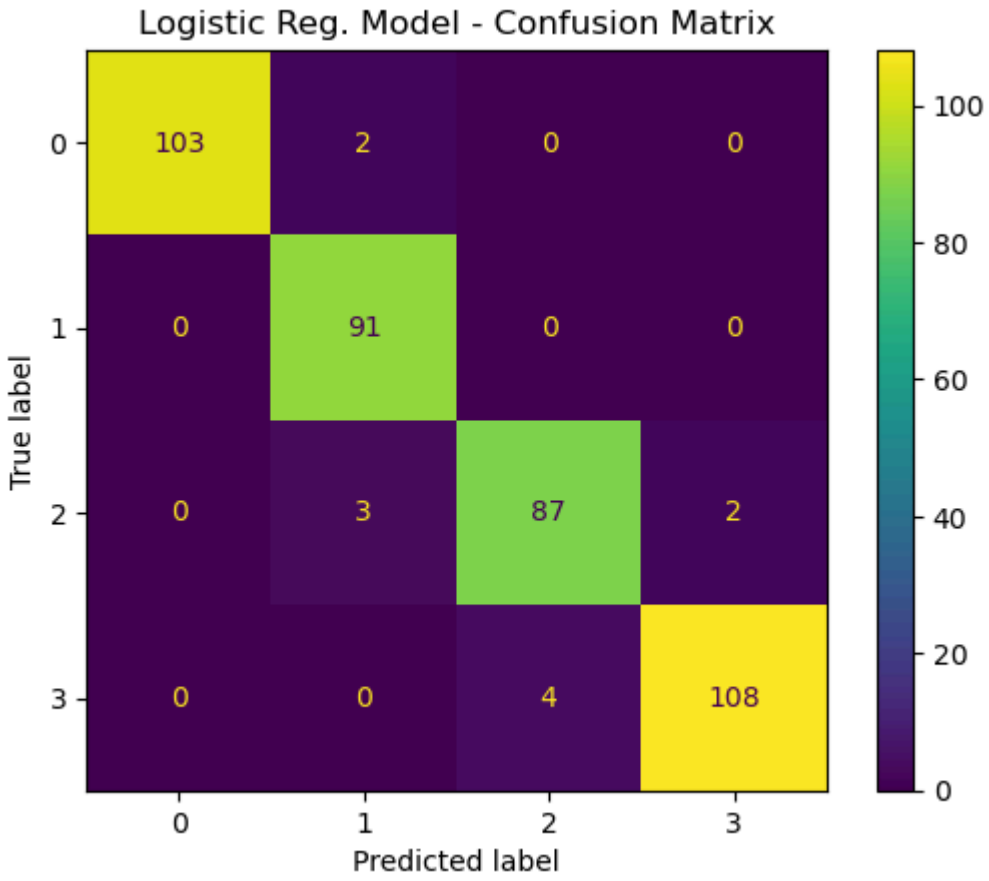
In [ ]:
```
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix
import matplotlib.pyplot as plt

# Create a logistic regression model using sklearn library
#clf=?
#clf.?
clf = LogisticRegression()
clf.fit(xtrain, ytrain)

#print score for test data
```

```
#print(?)
print(clf.score(xtest, ytest))
```

```
0.9725
```

In [ ]:
```
#cm = ?
#plt.title("Logistic Reg. Model - Confusion Matrix")
#plt.xticks(?, ?, fontsize=11)
#plt.yticks(?, ?, fontsize=11)
#plt.show()

cm = confusion_matrix(ytest, clf.predict(xtest), labels=clf.classes_)
cm_display = ConfusionMatrixDisplay(cm).plot()
cm_display.ax_.set_title("Logistic Reg. Model - Confusion Matrix")
```

Out[ ]:
```
Text(0.5, 1.0, 'Logistic Reg. Model - Confusion Matrix')
```



Q8: Train Logistic Regression Model using cross-validation on *xtrain, ytrain* data.

- Apply K fold cross validation technique for the model training (cross_val_score), and set K to 5 or 10.
- Print the different scores from different folds

A8: Replace ??? with code in the code cell below

In [ ]:
```
from sklearn.model_selection import cross_val_score

# Use sklearn for 5 fold cross validation
#scores_log= ?
scores_log_5 = cross_val_score(clf, xtrain, ytrain, cv=5)
```

```
scores_log_10 = cross_val_score(clf, xtrain, ytrain, cv=10)

# print the scores from different folds
#print(?)
print(scores_log_5)
print(scores_log_10)
```

```
[0.95625  0.953125 0.959375 0.95625  0.953125]
[0.93125 0.96875 0.96875 0.95    0.95    0.98125 0.975   0.93125 0.9375
 0.95625]
```

Classification Model 2: K Nearest Neighbor Classifier

Here, we learn how to fit KNN on the train dataset using k-fold cross validation, and evaluate its classification accuracy on the train dataset using confusion matrix.

Q9 Build a KNN Classification Model for the dataset as following:

1. Create a KNN model using sklearn library, and initialize n_neighbors as described in documentation.
2. Fit the model with the train data
3. Predict the values from test data
4. Print out the score from training and test data
5. Repeat Step 1.- 4. for a range of `n_neighbors` values (k in kNN) from 1 to 30.

A9 Replace ??? with code in the code cell below

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier


# Define KNN model
#for k in range(?,?):

#    knn = ?

    #Fit KNN model on xtrain, ytrain from above
#    knn.?
    #predict y values from xtest
#    y_pred=knn.?

    #print score for test data
#    print("K: ",k,"Train Score: ",knn.?, "Test Score: ",knn.?)

for k in range(1, 31): # this does 1 through 30 inclusive
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(xtrain, ytrain)
    y_pred = knn.predict(xtest)
    print(f'K: {k}  Train Score: {knn.score(xtrain, ytrain)}  Test Score: {knn.score(x
```

```
K: 1  Train Score: 1.0  Test Score: 0.8875
K: 2  Train Score: 0.933125  Test Score: 0.8675
K: 3  Train Score: 0.955  Test Score: 0.915
K: 4  Train Score: 0.92625  Test Score: 0.895
K: 5  Train Score: 0.930625  Test Score: 0.915
K: 6  Train Score: 0.92  Test Score: 0.91
K: 7  Train Score: 0.924375  Test Score: 0.91
K: 8  Train Score: 0.91875  Test Score: 0.9175
K: 9  Train Score: 0.92625  Test Score: 0.9225
K: 10  Train Score: 0.91875  Test Score: 0.9175
K: 11  Train Score: 0.9225  Test Score: 0.925
K: 12  Train Score: 0.92  Test Score: 0.9225
K: 13  Train Score: 0.924375  Test Score: 0.93
K: 14  Train Score: 0.92  Test Score: 0.91
K: 15  Train Score: 0.923125  Test Score: 0.915
K: 16  Train Score: 0.921875  Test Score: 0.915
K: 17  Train Score: 0.9225  Test Score: 0.915
K: 18  Train Score: 0.92  Test Score: 0.9025
K: 19  Train Score: 0.919375  Test Score: 0.9125
K: 20  Train Score: 0.9175  Test Score: 0.905
K: 21  Train Score: 0.91875  Test Score: 0.9
K: 22  Train Score: 0.92  Test Score: 0.9075
K: 23  Train Score: 0.916875  Test Score: 0.915
K: 24  Train Score: 0.92  Test Score: 0.9
K: 25  Train Score: 0.920625  Test Score: 0.9125
K: 26  Train Score: 0.92375  Test Score: 0.9125
K: 27  Train Score: 0.921875  Test Score: 0.915
K: 28  Train Score: 0.920625  Test Score: 0.9125
K: 29  Train Score: 0.91875  Test Score: 0.92
K: 30  Train Score: 0.913125  Test Score: 0.91
```

Q9 Part 2:

What is the best `n_neighbors`? Why?

A9 Write your part 2 answer in this cell

- There are likely many "best" options. Anything between k=5 and k=19 is likely a good contender. In this range, the training and test scores are reasonably close, not showing signs of overfitting or underfitting. I will pick 9.

Q10.

1. Create a KNN Classifier model using the best value of k found from previous question.
2. Train the model using xtrain, ytrain values.
3. Plot confusion matrix for the xtest and ytest, using ConfusionMatrixDisplay, see Visualization with Display Objects example.

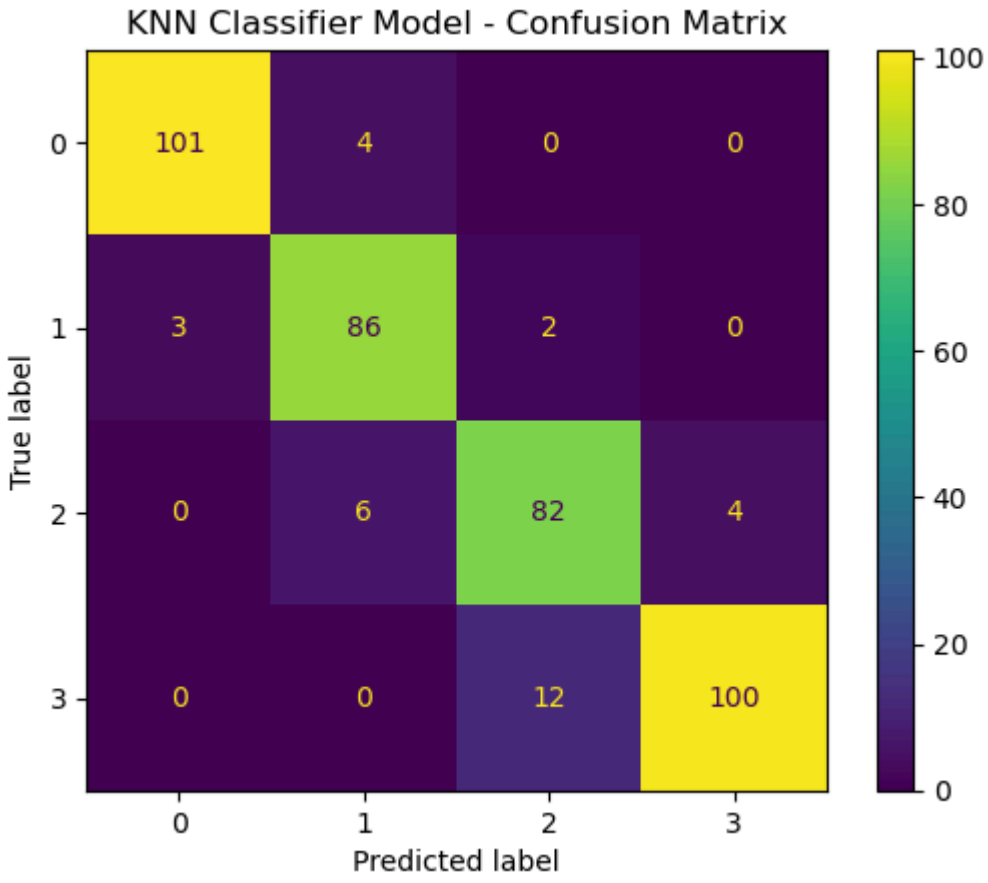
A10 Replace ??? with code in the code cell below

In [ ]:
```
knn_best = KNeighborsClassifier(n_neighbors=9)

knn_best.fit(xtrain, ytrain)
```

Out[ ]:
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=9)
```

In [ ]:
```
#cm = ?
#plt.figure()
#plot_confusion_matrix(cm, figsize=(12,8), hide_ticks=True, cmap=plt.cm.Blues)
#plt.title("KNN Classifier Model - Confusion Matrix")
#plt.xticks(?)
#plt.yticks(?)
#plt.show()

#cm = confusion_matrix(ytest, clf.predict(xtest), labels=clf.classes_)

cm = confusion_matrix(ytest, knn_best.predict(xtest), labels=knn_best.classes_)
cm_display = ConfusionMatrixDisplay(cm).plot()
cm_display.ax_.set_title("KNN Classifier Model - Confusion Matrix")
```

Out[ ]:
```
Text(0.5, 1.0, 'KNN Classifier Model - Confusion Matrix')
```



Q11 Train KNN classifier using cross-validation approach, sklearn.cross_validation tutorial.

Note:

Try a range of `n_neighbors` values (k in kNN) from 1 to 30.

A11 Replace ??? with code in the code cell below **

In [ ]:
```
# Define KNN model
from sklearn.model_selection import cross_val_score

for k in range(1 , 31): # for 1-30 inclusive
    #Define KNN model
    #knn_crossval = ?
    knn_crossval = KNeighborsClassifier(n_neighbors=k)
    # Use sklearn for 5 fold cross validation
    #scores_cv=?
    scores_cv = cross_val_score(knn_crossval, xtrain, ytrain, cv=5)
    # print the scores from different folds
    #print(?)
    print(scores_cv)
```

```
[0.834375 0.8375   0.865625 0.859375 0.865625]
[0.85    0.85    0.84375 0.84375 0.85   ]
[0.88125  0.878125 0.85     0.846875 0.85625 ]
[0.875    0.865625 0.846875 0.853125 0.85    ]
[0.8625   0.890625 0.8625   0.86875  0.85625 ]
[0.86875  0.890625 0.865625 0.878125 0.85    ]
[0.88125  0.8875   0.865625 0.865625 0.865625]
[0.8875   0.871875 0.88125  0.865625 0.859375]
[0.875    0.88125  0.890625 0.8625   0.878125]
[0.86875  0.890625 0.896875 0.871875 0.86875 ]
[0.859375 0.909375 0.903125 0.884375 0.86875 ]
[0.8625   0.8875   0.9125   0.878125 0.86875 ]
[0.875    0.890625 0.915625 0.875    0.88125 ]
[0.859375 0.903125 0.915625 0.8625   0.86875 ]
[0.875    0.903125 0.921875 0.871875 0.86875 ]
[0.875    0.9      0.909375 0.8625   0.871875]
[0.8875   0.915625 0.909375 0.878125 0.871875]
[0.8875   0.909375 0.896875 0.871875 0.88125 ]
[0.8875   0.915625 0.890625 0.875    0.884375]
[0.890625 0.90625  0.890625 0.875    0.86875 ]
[0.884375 0.925    0.89375  0.88125  0.8875  ]
[0.89375  0.915625 0.8875   0.878125 0.865625]
[0.88125  0.91875  0.890625 0.9      0.865625]
[0.896875 0.90625  0.884375 0.890625 0.8625  ]
[0.8875   0.915625 0.89375  0.90625  0.8625  ]
[0.884375 0.903125 0.884375 0.89375  0.875   ]
[0.890625 0.91875  0.90625  0.9      0.878125]
[0.884375 0.909375 0.890625 0.8875   0.88125 ]
[0.884375 0.909375 0.9      0.890625 0.890625]
[0.890625 0.90625  0.884375 0.884375 0.88125 ]
```

Comparison

Q12 Compare the two models (trained using xtrain,ytrain) in terms of score.

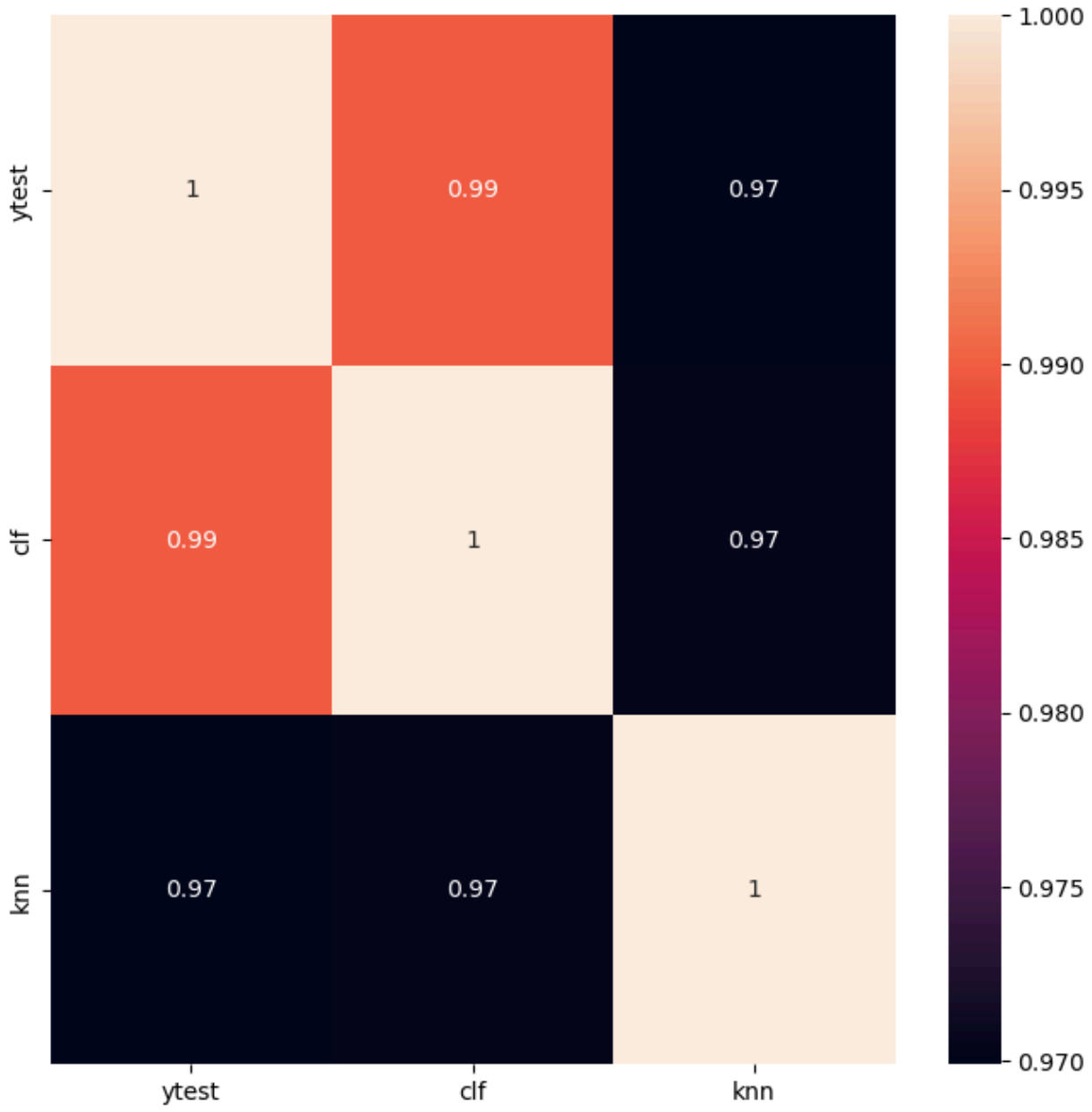
- Train two different models on Train data
- Predict xtest using the trained models
- Make a correlation matrix between ytest and predicted ytest values from the two Models
- Your resulting matrix should be `3x3 correlation matrix` for xtest, ytest data
  - The matrix is symmetric
  - It will provide the correlation between two models predictions plus ytest

- Hint: You can create a new dataframe using these values and use corr() function for creating the corelation matrix. Use meaningful column name while creating the dataframe.

A12 Replace ??? with code in the code cell below

In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Predict Train dataset using logistic reg
#clf= LogisticRegression()
#clf.?
#ypred_clf=?

# Predict Train dataset using KNN
#knn= KNeighborsClassifier(n_neighbors=29)
#knn.?
#ypred_knn=?

#print(ytest.shape, ypred_clf.shape, ypred_knn.shape)
# Create a dataframe using the predicted results from the models
#df = ?

#copute correlation

# Now use seaborn library to plot the heatmap correlation matrix
#plt.figure(figsize=(8,8))
#sns.?

clf = LogisticRegression()
clf.fit(xtrain, ytrain)
ypred_clf = clf.predict(xtest)

knn = KNeighborsClassifier(n_neighbors=29)
knn.fit(xtrain, ytrain)
ypred_knn = knn.predict(xtest)

print(ytest.shape, ypred_clf.shape, ypred_knn.shape)

df_clf_knn = pd.DataFrame({'ytest': ytest, 'clf': ypred_clf, 'knn': ypred_knn})
plt.figure(figsize=(8,8))
sns.heatmap(data=df_clf_knn.corr(), annot=True)
```

Out[ ]:
```
(400,) (400,) (400,)
<Axes: >
```


ytest
clf
knn
1
0.99
0.97
0.99
1
0.97
0.97
0.97
1
ytest
clf
knn
1.000
0.995
0.990
0.985
0.980
0.975
0.970