ara218 / Ara218 Private

Code Issues Pull requests Projects Wiki Security Insights Sett

main

Ara218 / HomeWork / HW_4.ipynb

ara218 adding homework 4 to git repo History

1 contributor

1267 lines (1267 sloc) | 515 KB

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 [1]:

```
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_
df = pd.read_csv("C:\\Users\\alsae\\Desktop\\fake\\2024Spring\\data\\mobile_dat
```

In [2]:

```
df.head()
```

Out[2]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobil
0	842	0	2.2	0	1	0	7	0.6	
1	1021	1	0.5	1	0	1	53	0.7	
2	563	1	0.5	1	2	1	41	0.9	
3	615	1	2.5	0	0	0	10	0.8	
4	1821	1	1.2	0	13	1	44	0.6	

5 rows × 21 columns

In [3]:

```
y = df['price_range']

x = df
```

In [4]:

```
df.head()
```

Out[4]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobil
0	842	0	2.2	0	1	0	7	0.6	
1	1021	1	0.5	1	0	1	53	0.7	
2	563	1	0.5	1	2	1	41	0.9	
3	615	1	2.5	0	0	0	10	0.8	

4	1821	1	1.2	0	13	1	44	0.6

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 [5]:

```
x.isnull().sum()
```

Out[5]:

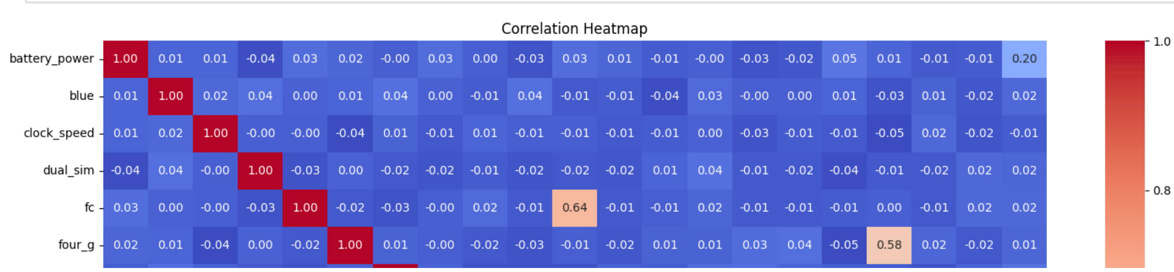
```
battery_power    0
blue             0
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w       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

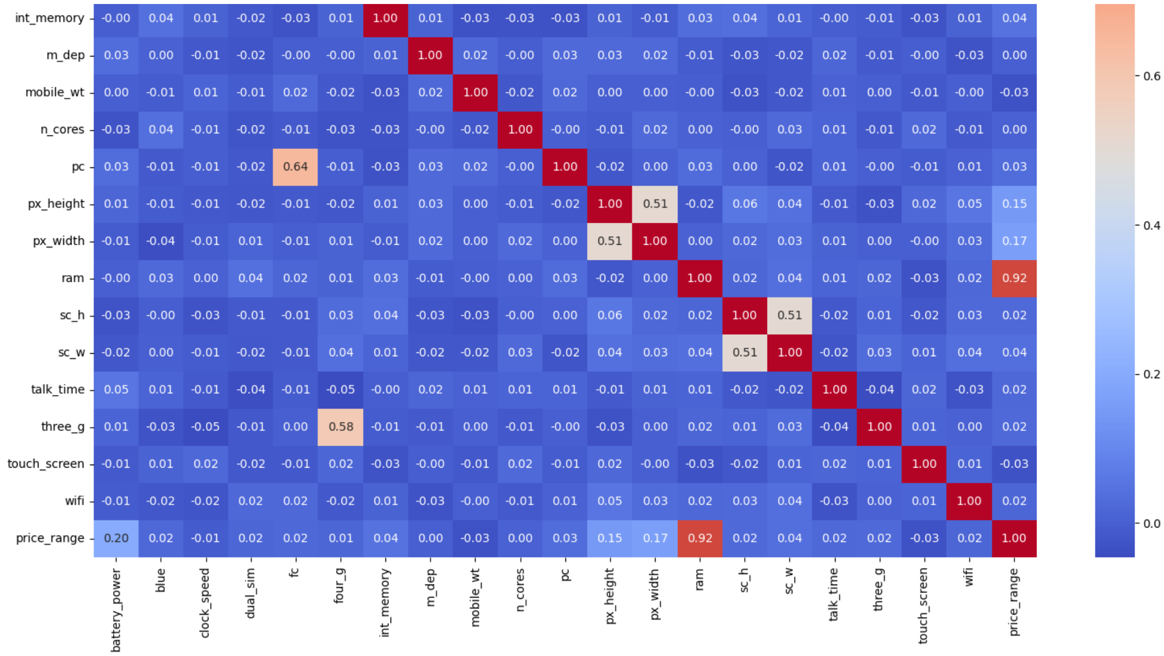
Q3.1 Use seaborn heatmap chart to visualize the correlations between the columns. Replace ??? with code in the code cell below

A3.1

In [6]:

```
import seaborn as sns
plt.figure(figsize=(18,12))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```





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 the columns that are a good indicatotor are battery power, px height, px width, and ram has a very strong correlation with .92

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 [7]:

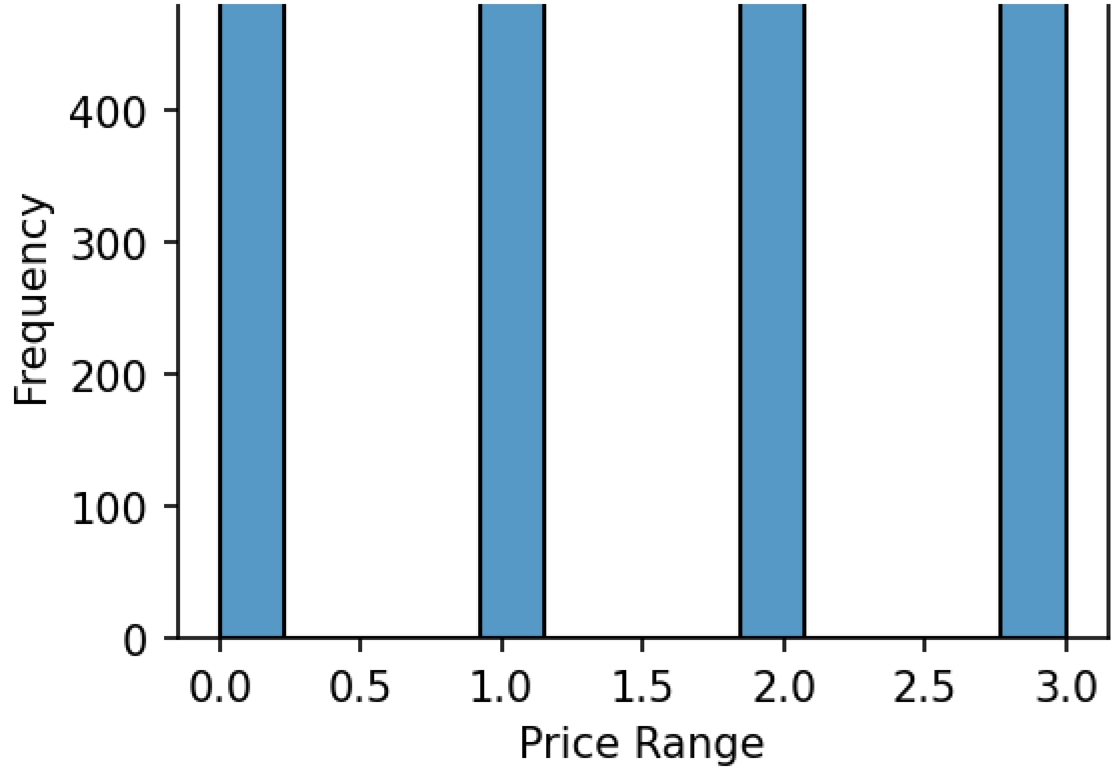
```
# A3 Part 3:
indicator_columns = ['battery_power', 'px_height', 'px_width', 'ram']
x = x[indicator_columns]
```

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

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

In [8]:

```
plt.figure(figsize=(4,3),dpi=150)
sns.histplot(data=df, x='price_range')

plt.title('Distribution of Price Range')
plt.xlabel('Price Range')
plt.ylabel('Frequency')
plt.show()
```



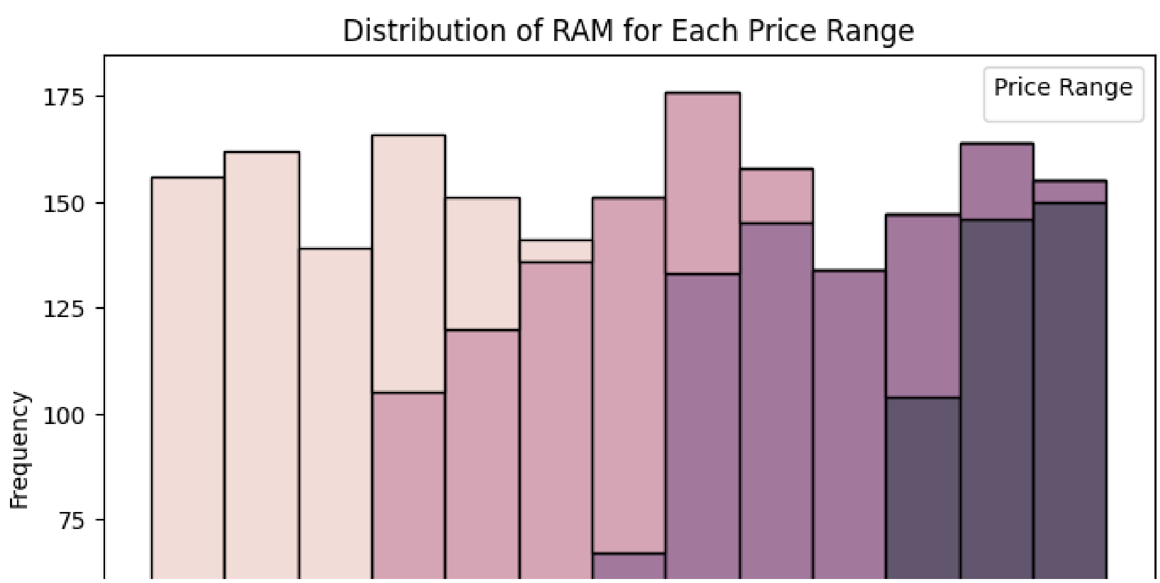
Q5: Use seaborn *histplot* to present the relation between *price_range* and the *ram* of a mobile

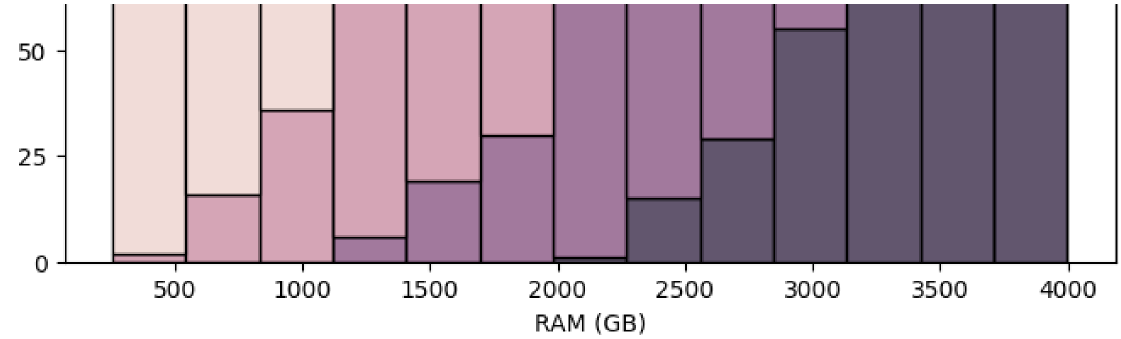
A5 Replace ??? with code in the code cell below

In [9]:

```
plt.figure(figsize=(8, 6))
sns.histplot(data=df, x='ram', hue='price_range', multiple='stack')
plt.title('Distribution of RAM for Each Price Range')
plt.xlabel('RAM (GB)')
plt.ylabel('Frequency')
plt.legend(title='Price Range')
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label sta
rt with an underscore are ignored when legend() is called with no argument.
```





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 [10]:
```
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)
```

In [11]:
```
xtrain, xtest, ytrain, ytest = train_test_split(x_scaled, y, test_size=0.2, ran
```

In [12]:
```
print("Number of instances in the train set:", xtrain.shape[0])
print("Number of instances in the test set:",xtest.shape[0])
```

```
Number of instances in the train set: 1600
Number of instances in the test set: 400
```

In [13]:
```
ytrain.value_counts()
```

Out[13]:
```
1    409
2    408
0    395
3    388
Name: price_range, dtype: int64
```

In [14]:
```
ytest.value_counts()
```

Out[14]:
```
3    112
0    105
2     92
1     91
Name: price_range, dtype: int64
```

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.

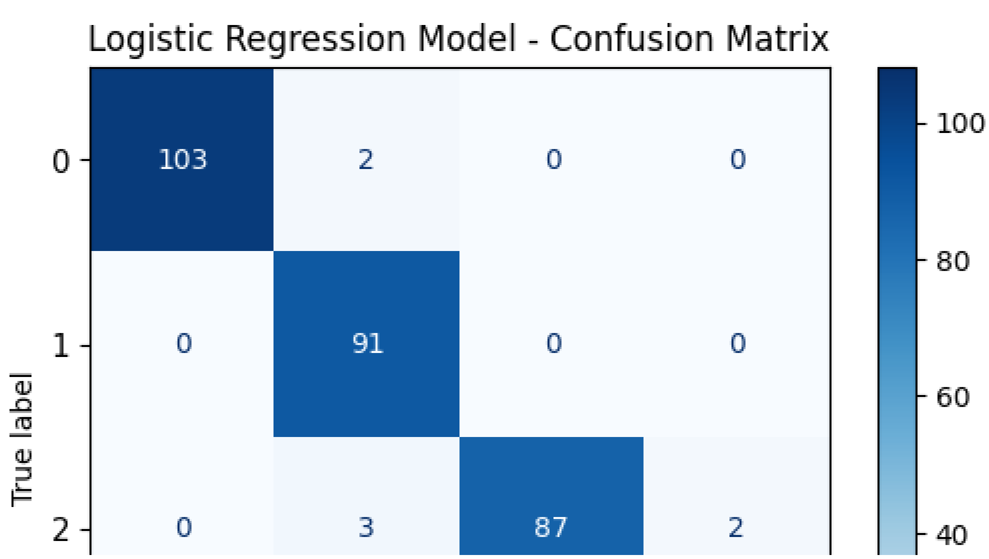
A7 Replace ??? with code in the code cell below

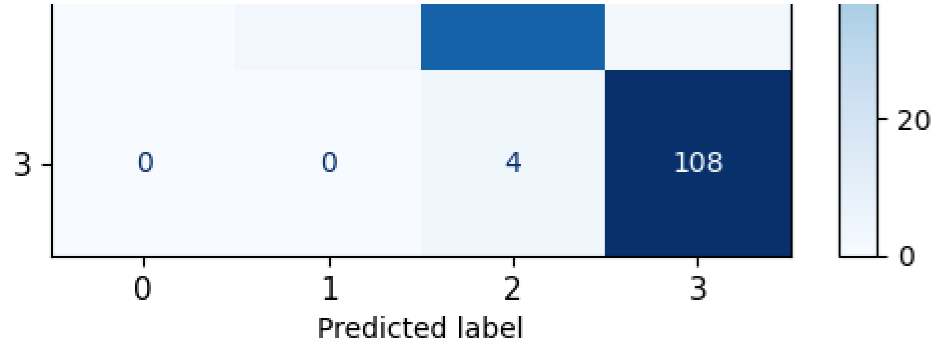
In [15]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDi
import matplotlib.pyplot as plt


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

#print score for test data
score = clf.score(xtest, ytest)
print("Accuracy score:", score)
```

```
Accuracy score: 0.9725
```

In [16]:
```
ypred = clf.predict(xtest)
```

In [17]:
```
cm = confusion_matrix(ytest, ypred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=clf.classes_)
disp.plot(cmap=plt.cm.Blues)
plt.title("Logistic Regression Model - Confusion Matrix")
plt.xticks(fontsize=11)
plt.yticks(fontsize=11)
plt.show()
```





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 [18]:

```
from sklearn.model_selection import cross_val_score

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

# print the scores from different folds
print(scores_log)
```

```
[0.95625  0.953125 0.959375 0.95625  0.953125]
```

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 [19]:

```
from sklearn.neighbors import KNeighborsClassifier
```

```
# Define KNN model
for k in range(1,31):

    knn = KNeighborsClassifier(n_neighbors=k)

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

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

```
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 the best number of neighbor is 13, during training the knn with 31 the one that had the highest test score was 13

Q10.

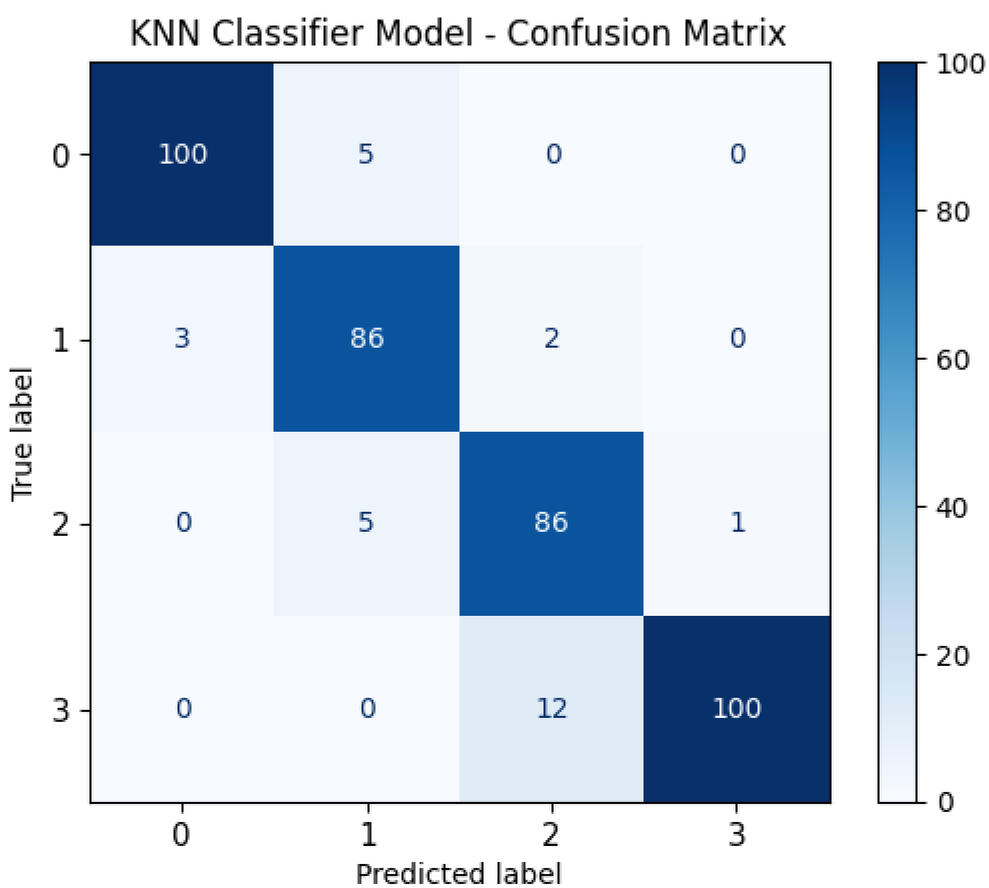
1. Create a KNN Classifier model using the best value of k found from previous question.
2. Train the model using xtrain, ytrain values.
3. Plot confusion matrix for the xtest and ytest, using ConfusionMatrixDisplay, see Visualization with Display Objects example.

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

In [20]:
```
knn_best = 13

knn_best = KNeighborsClassifier(n_neighbors=knn_best)
```

In [21]:
```
knn_best.fit(xtrain, ytrain)

y_pred_best = knn_best.predict(xtest)
```

In [22]:
```
cm = confusion_matrix(ytest, y_pred_best)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=knn_best.clas
disp.plot(cmap=plt.cm.Blues)
plt.title("KNN Classifier Model - Confusion Matrix")
plt.xticks(fontsize=11)
plt.yticks(fontsize=11)
plt.show()
```



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 [23]:

```python
# Define KNN model
from sklearn.model_selection import cross_val_score

for k in range(1 , 31):
    #Define KNN model
    knn_crossval = KNeighborsClassifier(n_neighbors=k)

    # Use sklearn for 5 fold cross validation
    scores_cv=cross_val_score(knn_crossval,xtrain,ytrain,cv=5)

    # print the scores from different folds
    print("K:", k, "Cross-Validation Scores:", scores_cv)
```

```
K: 1 Cross-Validation Scores: [0.834375 0.8375   0.865625 0.859375 0.865625]
K: 2 Cross-Validation Scores: [0.85    0.85    0.84375 0.84375 0.85    ]
K: 3 Cross-Validation Scores: [0.88125  0.878125 0.85     0.846875 0.85625 ]
K: 4 Cross-Validation Scores: [0.875    0.865625 0.846875 0.853125 0.85    ]
K: 5 Cross-Validation Scores: [0.8625   0.890625 0.8625   0.86875  0.85625 ]
K: 6 Cross-Validation Scores: [0.86875  0.890625 0.865625 0.878125 0.85    ]
K: 7 Cross-Validation Scores: [0.88125  0.8875   0.865625 0.865625 0.865625]
K: 8 Cross-Validation Scores: [0.8875   0.871875 0.88125  0.865625 0.859375]
K: 9 Cross-Validation Scores: [0.875    0.88125  0.890625 0.8625   0.878125]
K: 10 Cross-Validation Scores: [0.86875  0.890625 0.896875 0.871875 0.86875 ]
K: 11 Cross-Validation Scores: [0.859375 0.909375 0.903125 0.884375 0.86875 ]
K: 12 Cross-Validation Scores: [0.8625   0.8875   0.9125   0.878125 0.86875 ]
K: 13 Cross-Validation Scores: [0.875    0.890625 0.915625 0.875    0.88125 ]
K: 14 Cross-Validation Scores: [0.859375 0.903125 0.915625 0.8625   0.86875 ]
K: 15 Cross-Validation Scores: [0.875    0.903125 0.921875 0.871875 0.86875 ]
K: 16 Cross-Validation Scores: [0.875    0.9      0.909375 0.8625   0.871875]
K: 17 Cross-Validation Scores: [0.8875   0.915625 0.909375 0.878125 0.871875]
K: 18 Cross-Validation Scores: [0.8875   0.909375 0.896875 0.871875 0.88125 ]
K: 19 Cross-Validation Scores: [0.8875   0.915625 0.890625 0.875    0.884375]
K: 20 Cross-Validation Scores: [0.890625 0.90625  0.890625 0.875    0.86875 ]
K: 21 Cross-Validation Scores: [0.884375 0.925    0.89375  0.88125  0.8875  ]
K: 22 Cross-Validation Scores: [0.89375  0.915625 0.8875   0.878125 0.865625]
K: 23 Cross-Validation Scores: [0.88125  0.91875  0.890625 0.9      0.865625]
K: 24 Cross-Validation Scores: [0.896875 0.90625  0.884375 0.890625 0.8625  ]
K: 25 Cross-Validation Scores: [0.8875   0.915625 0.89375  0.90625  0.8625  ]
K: 26 Cross-Validation Scores: [0.884375 0.903125 0.884375 0.89375  0.875   ]
K: 27 Cross-Validation Scores: [0.890625 0.91875  0.90625  0.9      0.878125]
K: 28 Cross-Validation Scores: [0.884375 0.909375 0.890625 0.8875   0.88125 ]
K: 29 Cross-Validation Scores: [0.884375 0.909375 0.9      0.890625 0.890625]
K: 30 Cross-Validation Scores: [0.890625 0.90625  0.884375 0.884375 0.88125 ]
```

Comparison

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

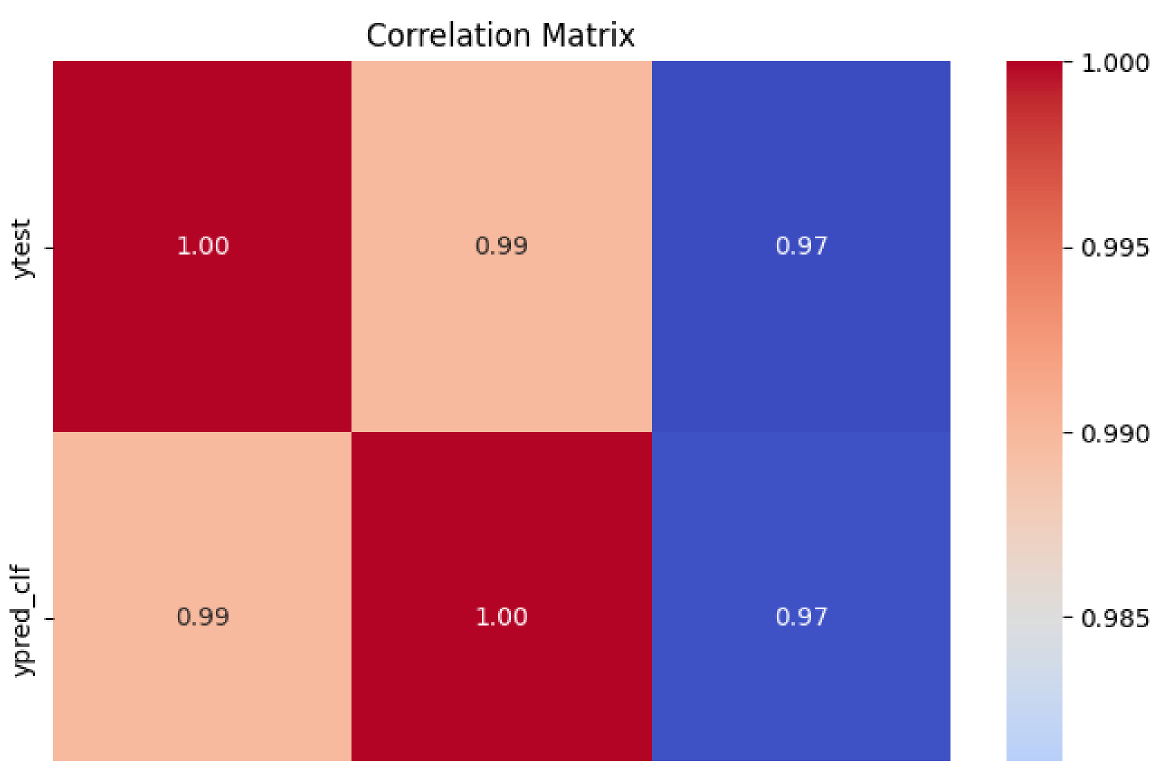
- Train two different models on Train data
- Predict xtest using the trained models
- Make a correlation matrix between ytest and predicted ytest values from the two Models
- Your resulting matrix should be `3x3 correlation matrix` for xtest, ytest data
  - The matrix is symmetric

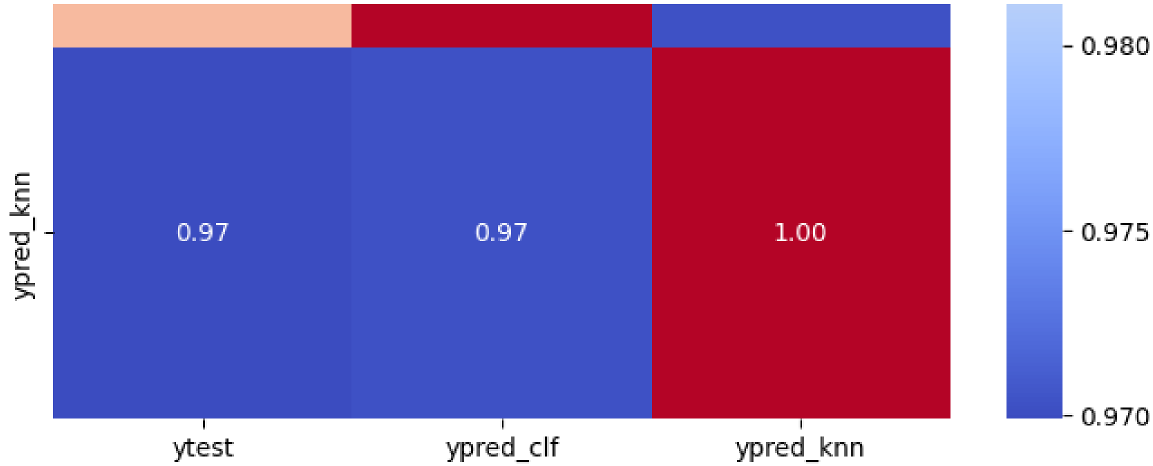
- It will provide the correlation between two models predictions plus ytest
- Hint: You can create a new dataframe using these values and use corr() function for creating the corelation matrix. Use meaningful column name while creating the dataframe.

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

In [24]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Predict Train dataset using logistic reg
clf= LogisticRegression()
clf.fit(xtrain,ytrain)
ypred_clf= clf.predict(xtest)

# Predict Train dataset using KNN
knn= KNeighborsClassifier(n_neighbors=29)
knn.fit(xtrain,ytrain)
ypred_knn= knn.predict(xtest)

print(ytest.shape, ypred_clf.shape, ypred_knn.shape)
# Create a dataframe using the predicted results from the models
df = pd.DataFrame({'ytest': ytest, 'ypred_clf': ypred_clf, 'ypred_knn': ypred_k

#copute correlation
correlation_matrix = df.corr()

# Now use seaborn library to plot the heatmap correlation matrix
plt.figure(figsize=(8,8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```

```
(400,) (400,) (400,)
```




ypred_knn
0.97
0.97
1.00
ytest
ypred_clf
ypred_knn
0.980
0.975
0.970