Supervised machine learning using KNN and logistic regression

Problem Statement

To study and analyze a dataset containing details of smartphone features and their relationships for predicting its price using algorithms such as Logistic regression and K Nearest Neighbors.

Dataset

https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification?datasetId=11167

In [1]:
```
#importing all necessary packages for analysis.
import pandas as pds
import numpy as npy
import seaborn as sbn
%matplotlib inline
import matplotlib.pyplot as plot
from sklearn.metrics import accuracy_score
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import GridSearchCdata_source = pds.read_csv("train.csv")
```

Exploring the data

In [2]:
```
#DataSource type
type(data_source)
```

Out[2]:
```
pandas.core.frame.DataFrame
```

In [3]:
```
#Retrieves top 5 columns from dataset
data_source.head()
```

Out[3]:

	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 [4]:
```
data_source.describe()
```

Out[4]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	2C
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	
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 [5]:
```
#Checks for duplicate values in the dataset
data_source.duplicated().sum()
```

Out[5]:
```
0
```

In [76]:
```
data_source.isna().sum()
```

Out[76]:
```
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
```

In [6]:
```
#Shares Info about the dataset
data_source.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   battery_power  2000 non-null   int64  
 1   blue           2000 non-null   int64  
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64  
 4   fc             2000 non-null   int64  
 5   four_g         2000 non-null   int64  
 6   int_memory     2000 non-null   int64  
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64  
 9   n_cores        2000 non-null   int64  
 10  pc             2000 non-null   int64  
 11  px_height      2000 non-null   int64  
 12  px_width       2000 non-null   int64  
 13  ram            2000 non-null   int64  
 14  sc_h           2000 non-null   int64  
 15  sc_w           2000 non-null   int64  
 16  talk_time      2000 non-null   int64  
 17  three_g        2000 non-null   int64  
 18  touch_screen   2000 non-null   int64  
 19  wifi           2000 non-null   int64  
 20  price_range    2000 non-null   int64  
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
```

In [7]:
```
print('trian_ShapeSize:',data_source.shape)
```

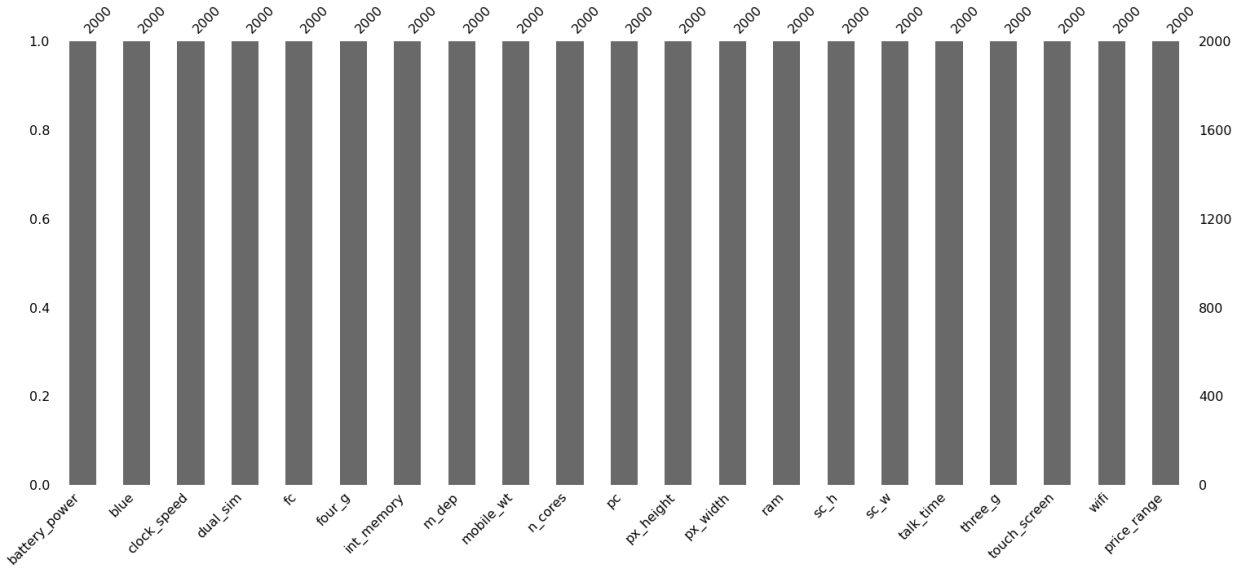
```
trian_ShapeSize: (2000, 21)
```

Preprocessing the data

In [8]:
```
#Installed pandas missingno package for data preprocessing and choosing the relevant f
!pip install missingno
```

```
Requirement already satisfied: missingno in c:\users\lenovo\anaconda3\lib\site-packag
es (0.5.2)
Requirement already satisfied: numpy in c:\users\lenovo\anaconda3\lib\site-packages
(from missingno) (1.21.5)
Requirement already satisfied: matplotlib in c:\users\lenovo\anaconda3\lib\site-packa
ges (from missingno) (3.5.1)
Requirement already satisfied: seaborn in c:\users\lenovo\anaconda3\lib\site-packages
(from missingno) (0.11.2)
Requirement already satisfied: scipy in c:\users\lenovo\anaconda3\lib\site-packages
(from missingno) (1.7.3)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\lenovo\anaconda3\lib\sit
e-packages (from matplotlib->missingno) (1.3.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\lenovo\anaconda3\lib
\site-packages (from matplotlib->missingno) (2.8.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\lenovo\anaconda3\lib\site-pa
ckages (from matplotlib->missingno) (9.0.1)
Requirement already satisfied: cycler>=0.10 in c:\users\lenovo\anaconda3\lib\site-pac
kages (from matplotlib->missingno) (0.11.0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\lenovo\anaconda3\lib\site
-packages (from matplotlib->missingno) (3.0.4)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\lenovo\anaconda3\lib\sit
e-packages (from matplotlib->missingno) (4.25.0)
Requirement already satisfied: packaging>=20.0 in c:\users\lenovo\anaconda3\lib\site-
packages (from matplotlib->missingno) (21.3)
Requirement already satisfied: six>=1.5 in c:\users\lenovo\anaconda3\lib\site-package
s (from python-dateutil>=2.7->matplotlib->missingno) (1.16.0)
Requirement already satisfied: pandas>=0.23 in c:\users\lenovo\anaconda3\lib\site-pac
kages (from seaborn->missingno) (1.4.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\lenovo\anaconda3\lib\site-pac
kages (from pandas>=0.23->seaborn->missingno) (2021.3)
```

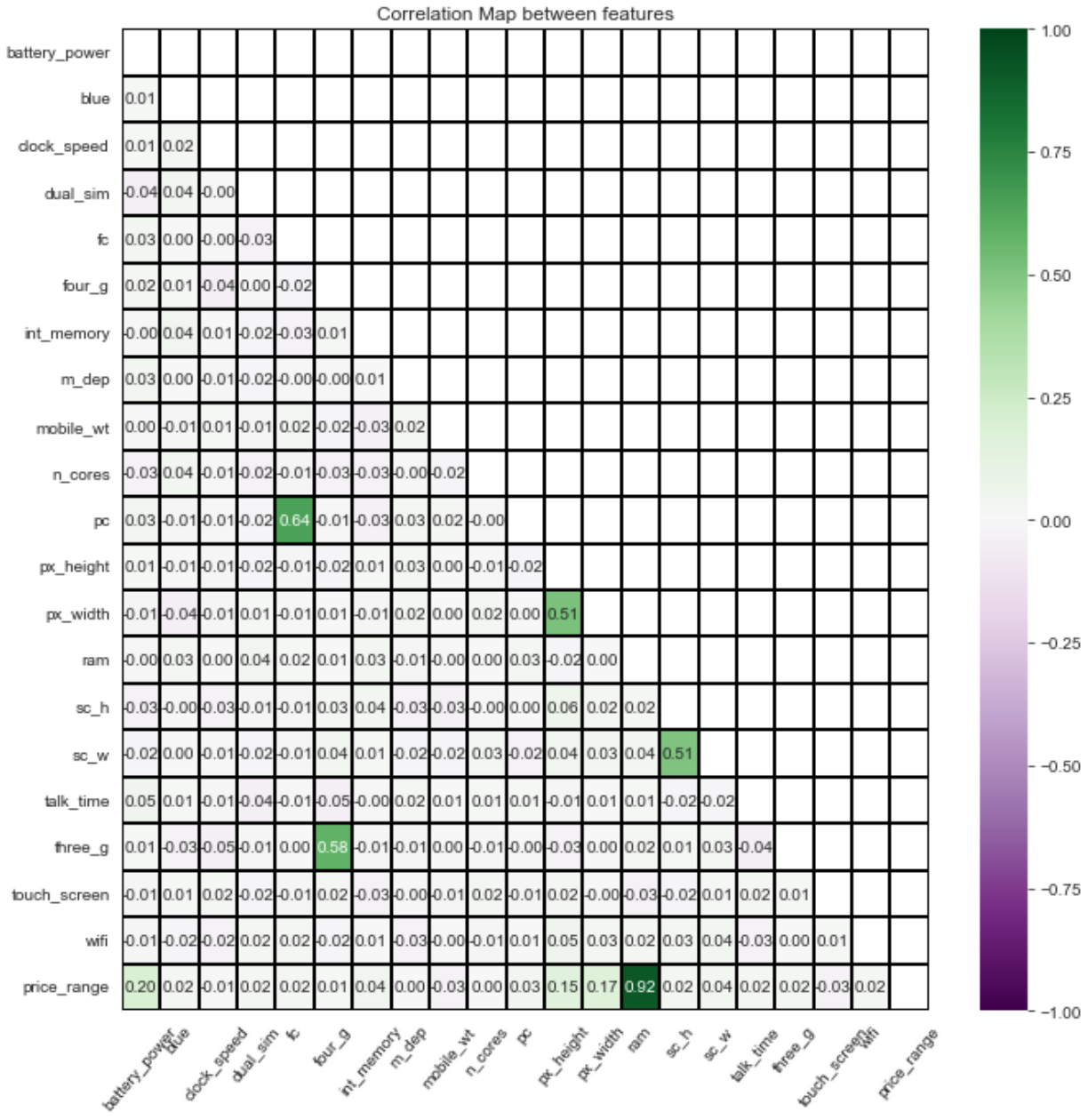
In [10]:
```
#Plot showing there are no missing values in DataSet
import missingno as msno
msno.bar(data_source)
plot.show()
```



Mapping Price range and features in the dataset

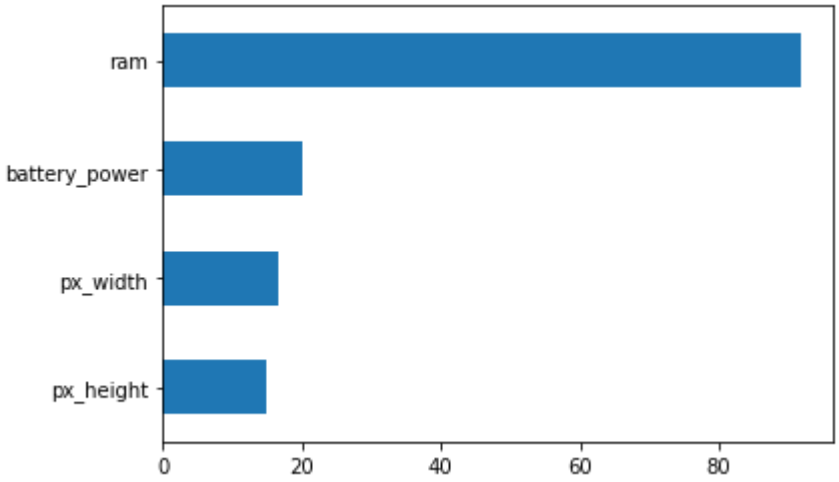
In [95]:
```
#Plotting a Map showing Correlation between price range and features
Correlationmatrix = npy.triu(data_source.corr())
```

```
sbn.set_style("white")
f,ax=plot.subplots(figsize = (11,11))
sbn.heatmap(data_source.corr(),annot= True,fmt = ".2f",ax=ax,
            vmin = -1,
            vmax = 1, mask = Correlationmatrix,cmap = "PRGn",
            linewidth = 0.3,linecolor = "black")
plot.xticks(rotation=50)
plot.yticks(rotation=0)
plot.title('Correlation Map between features', size = 12)
plot.show()
```



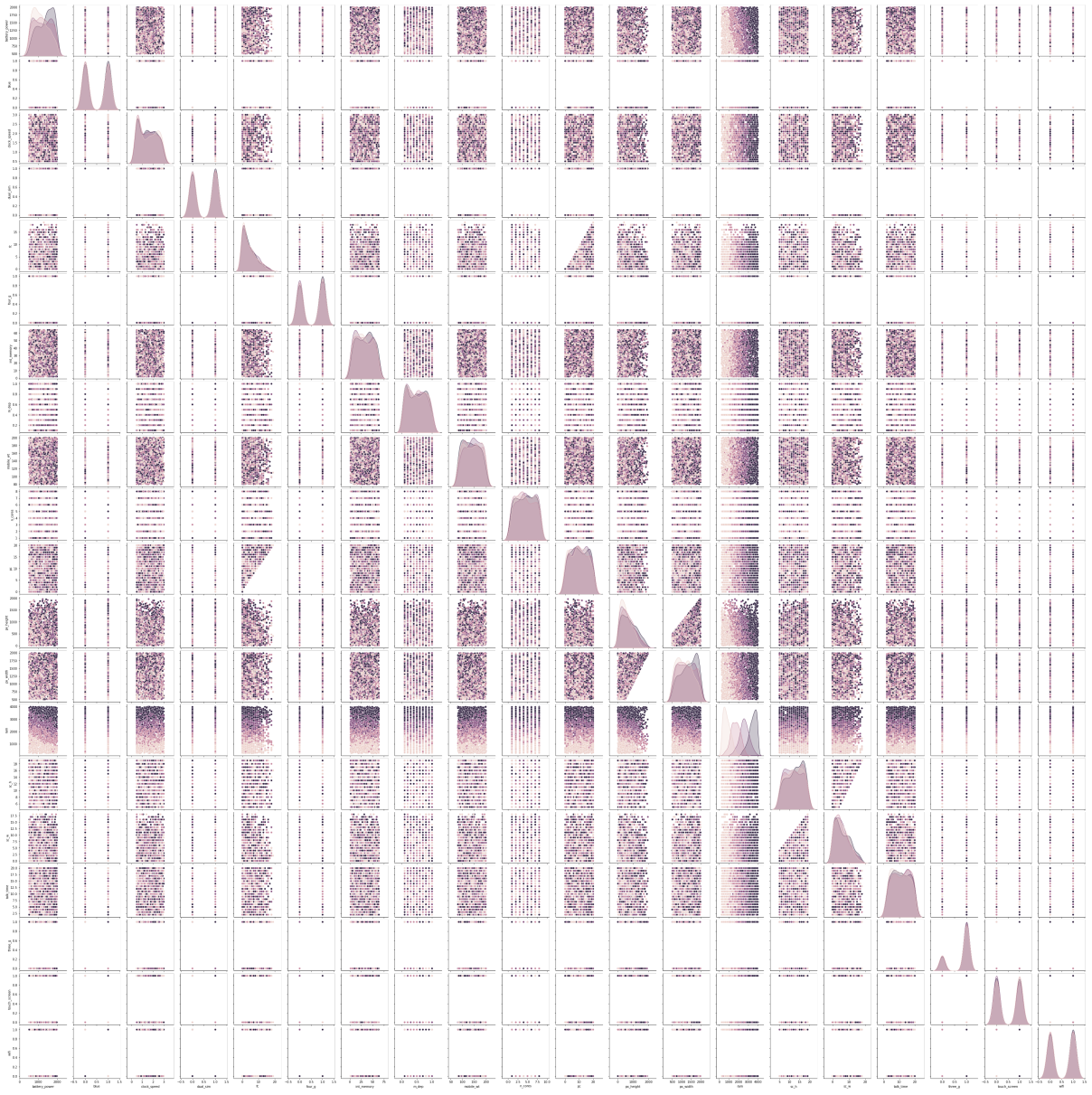
Analysing the early findings from the dataset

In [47]:
```
#Determining the Features influencing overall price of a smartphone.
(data_source.corr()["price_range"]* 100).sort_values(ascending=True).tail(5).drop("pri
plot.show()
```

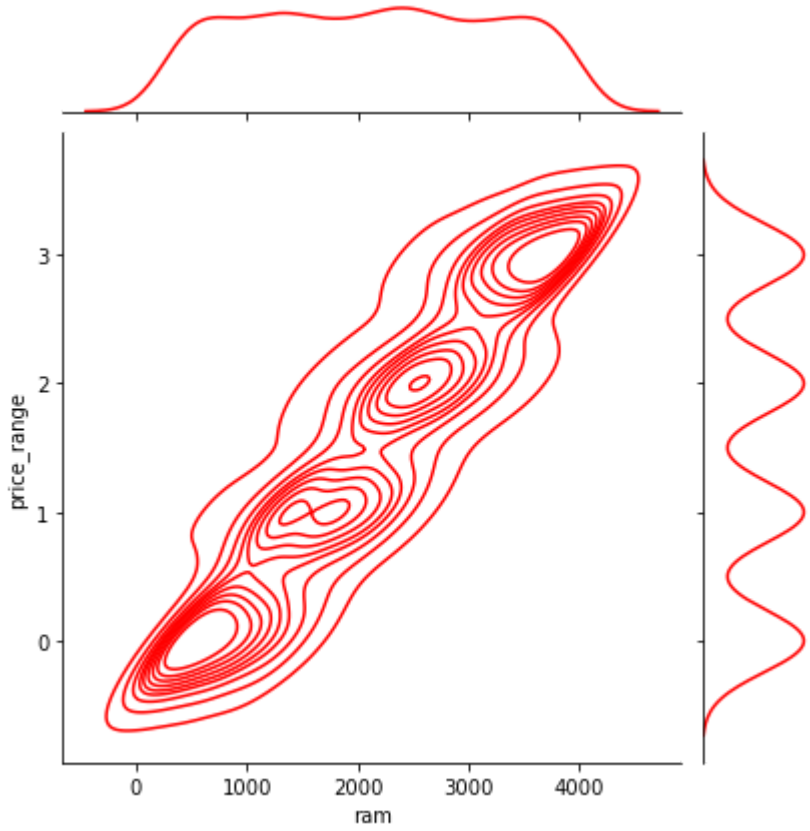


In [48]:
```
sbn.pairplot(data_source,hue='price_range')
```

Out[48]:
```
<seaborn.axisgrid.PairGrid at 0x14ec7c15040>
```



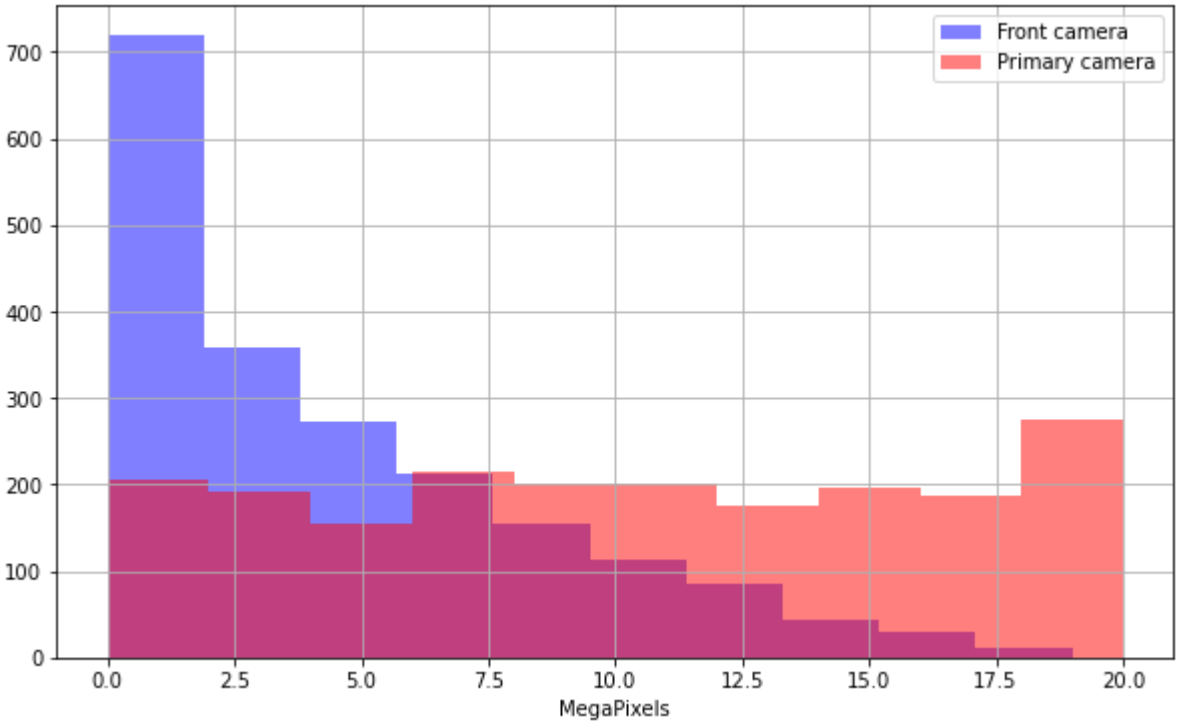
In [49]:
```
#Plotting the Effect of RAM on price range.
sbn.jointplot(x='ram',y='price_range',data=data_source,color='red',kind='kde');
```



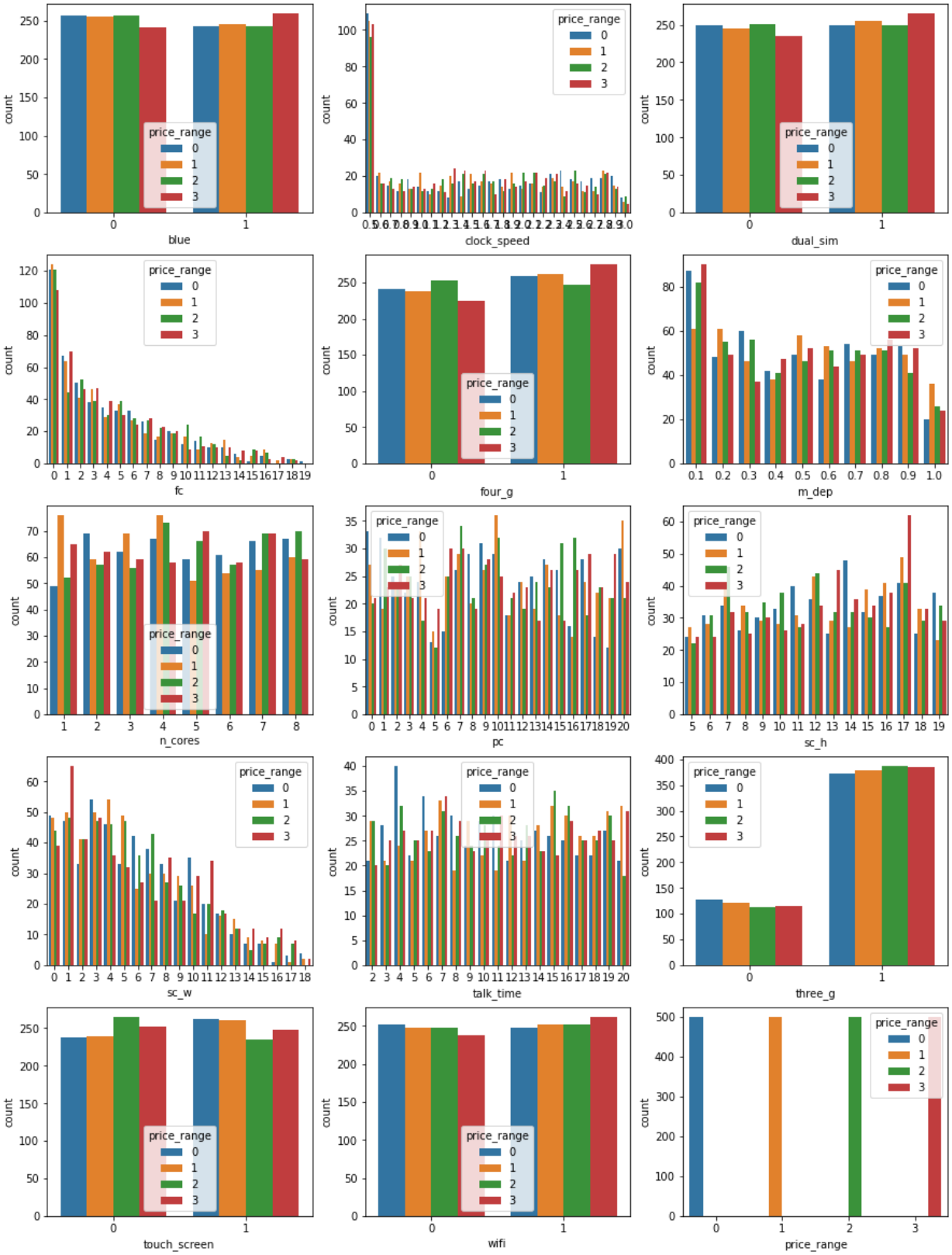
In [52]:
```
#Iterating over column to identify numerical and categorical features.
category_featrs=list()
numerical_featrs=list()
for c_name in data_source.columns:
    uniq_val = len(data_source[c_name].unique())
    if uniq_val<30:
      category_featrs.append(c_name)
    else:
      numerical_featrs.append(c_name)
```

In [53]:
```
#Plotting the Camera features influencing the price of mobile device
plot.figure(figsize=(10,6))
data_source['fc'].hist(alpha=0.5,color='blue',label='Front camera')
data_source['pc'].hist(alpha=0.5,color='red',label='Primary camera')
plot.legend()
plot.xlabel('MegaPixels')
```

Out[53]:
```
Text(0.5, 0, 'MegaPixels')
```

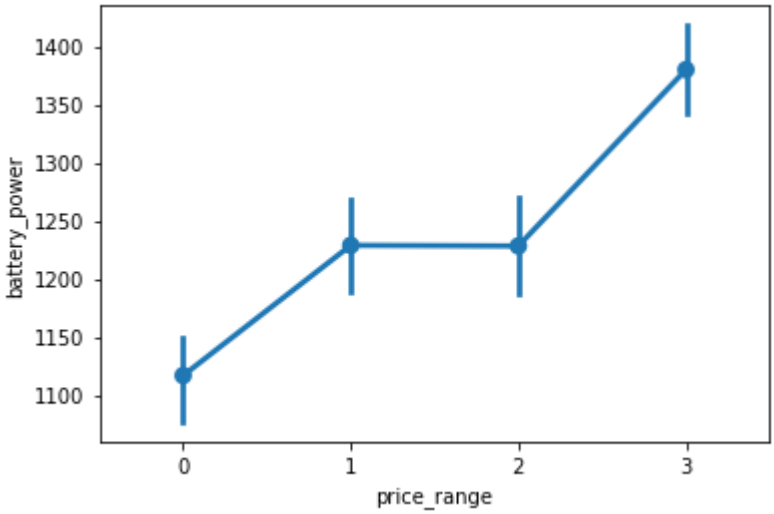


In [54]:
```
i = 1
plot.figure(figsize = (15,25))
for feature in category_featrs:
    plot.subplot(6,3,i)
    sbn.countplot(x = feature , data = data_source,hue='price_range')
    i +=1
```



In [56]:
```
#Impact of Lithium ion batteries on price range
sbn.pointplot(x='price_range',y='battery_power',data=data_source)
```

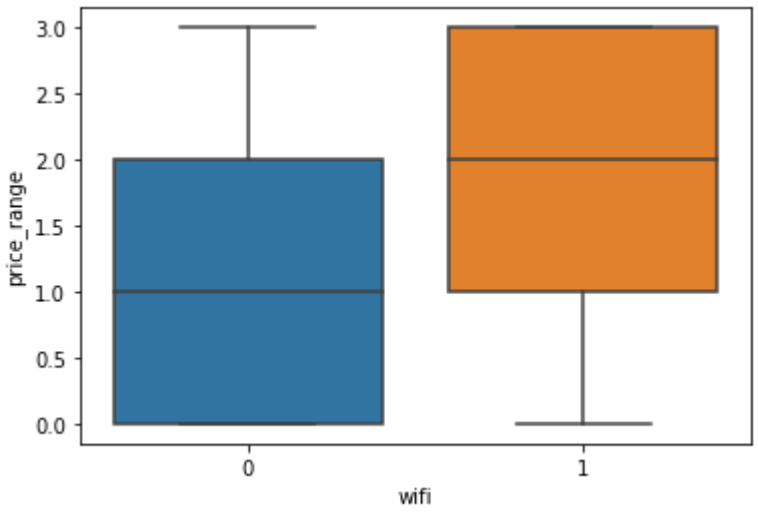
Out[56]:
```
<AxesSubplot:xlabel='price_range', ylabel='battery_power'>
```



In [58]:
```
#Plotting the effect of Wifi Modem on price range
sbn.boxplot(data_source['wifi'],data_source['price_range'])
```

```
C:\Users\LENOVO\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:

Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
```

Out[58]:
```
<AxesSubplot:xlabel='wifi', ylabel='price_range'>
```

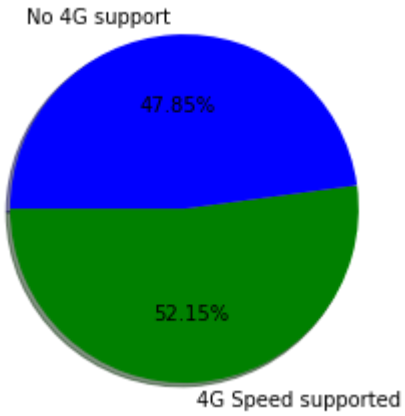


In [60]:
```
px.box(data_source,x='price_range',y='ram',
       color='price_range',template='ggplot2',
       labels={'price_range':'Product Price',
               'ram':'RAM size'},
       title="<b>Relationship between Ram and Price")
```

In [61]:
```
X=data_source.drop('price_range',axis=1)
```

In [62]:
```
y=data_source['price_range']
```

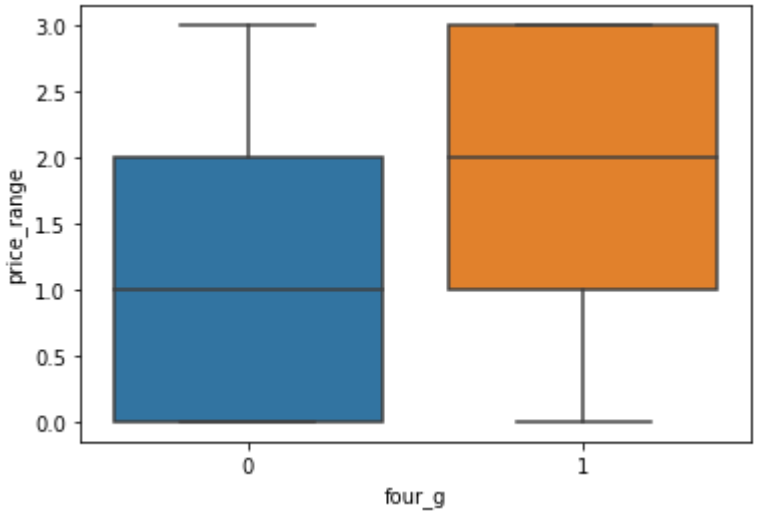
In [63]:
```
#Plotting the effect of 4G Modem on price range
lbls = ["4G Speed supported",'No 4G support']
val = data_source['four_g'].value_counts().values
fig1, ax1 = plot.subplots()
colors = ['green', 'blue']
ax1.pie(val, labels=lbls, autopct='%1.2f%%',shadow=True,startangle=180,colors=colors)
plot.show()
```



In [64]:
```
sbn.boxplot(data_source['four_g'],data_source['price_range'])
```

```
C:\Users\LENOVO\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:

Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
```

Out[64]:
```
<AxesSubplot:xlabel='four_g', ylabel='price_range'>
```



In [65]:
```
#Splitting the data set as 70 % training and 30% testing set
from sklearn.model_selection import train_test_split
```

In [66]:
```
X_trainData, X_testData, y_trainData, y_testData = train_test_split(X, y, test_size=0.
```

Training the K Nearest Neighbours model

In [67]:
```
#Modelling KNN with K value as 10
from sklearn.neighbors import KNeighborsClassifier
knnc = KNeighborsClassifier(n_neighbors=10)
knnc.fit(X_trainData,y_trainData)
```

Out[67]:
```
KNeighborsClassifier(n_neighbors=10)
```

In [68]:
```
#knnc.score(X_testData,y_testData)
```

In [69]:
```
#Finding the relationship between k value and error rate
error_rate = []
for i in range(1,20):
    knnc = KNeighborsClassifier(n_neighbors=i)
    knnc.fit(X_trainData,y_trainData)
    pred_i = knnc.predict(X_testData)
    error_rate.append(npy.mean(pred_i != y_testData))
```

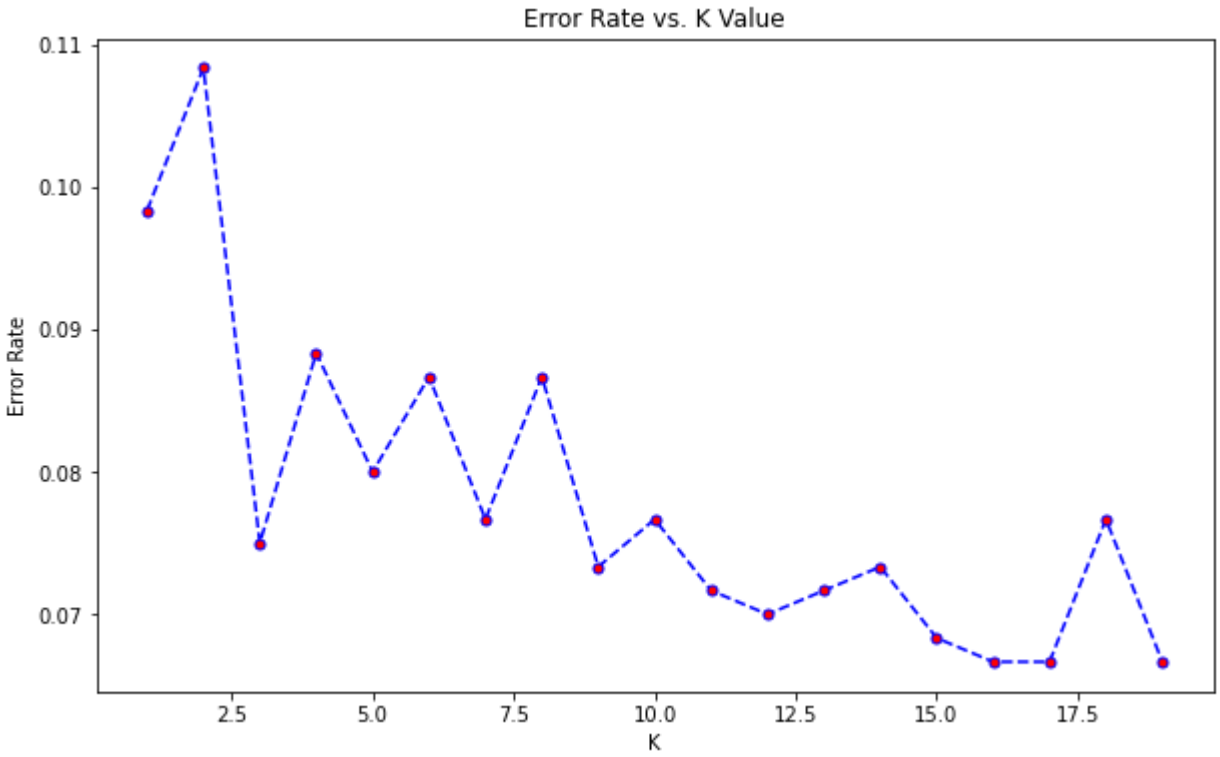
In [70]:
```
#calculating accuracy for KNN model
y_test_prediction = knnc.predict(X_testData)
y_train_prediction=knnc.predict(X_trainData)

knn_accscore=accuracy_score(y_test_prediction,y_testData)
knn_accscore
```

Out[70]:
```
0.9333333333333333
```

Finding the relationship between k value and error rate

In [72]:
```
#relationship between k value and error rate
import matplotlib.pyplot as plot
plot.figure(figsize=(10,6))
plot.plot(range(1,20),error_rate,color='blue', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=5)
plot.title('Error Rate vs. K Value')
plot.xlabel('K')
plot.ylabel('Error Rate')
```

Out[72]:
```
Text(0, 0.5, 'Error Rate')
```



KNN accuracy calculation

In [100…
```
#accuracy of train and test dataset for KNN model
print("Accuracy of Train DataSet:"+str(accuracy_score(y_train_prediction,y_trainData)*
print("Accuracy of Test DataSet:"+str(accuracy_score(y_test_prediction,y_testData)*100
```

```
Accuracy of Train DataSet:94.85714285714286
Accuracy of Test DataSet:93.33333333333333
```

Training the logistic regression model

In [78]:
```
#Scaling the data for logistic regression model
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
X_train_s = ss.fit_transform(X_trainData)
X_test_s = ss.transform(X_testData)
```

In [79]:
```
#Learning the dataset using Logisitic regression model
from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression()
```

In [80]:
```
logmodel.fit(X_train_s,y_trainData)
```

Out[80]:
```
LogisticRegression()
```

Accuracy Calculation

In [81]:
```
#calculating accuracy for logistic regression model
y_prediction = logmodel.predict(X_train_s)
```

```
accuracy_score(y_trainData, y_prediction)
```

Out[81]:
```
0.9771428571428571
```

In [82]:
```
y_predtest= logmodel.predict(X_test_s)
lr_accscore=accuracy_score(y_testData, y_predtest)
lr_accscore
```

Out[82]:
```
0.945
```

In [83]:
```
logmodel.score(X_test_s,y_testData)
```

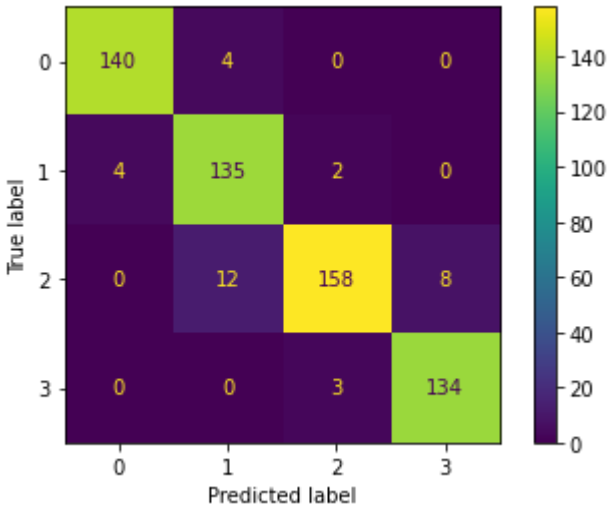
Out[83]:
```
0.945
```

Interpreting the results using Confusion matrix

In [84]:
```
#Plotting the confusion matrix for Logistic regression
import sklearn.metrics as metrics
metrics.plot_confusion_matrix(logmodel,X_test_s, y_testData)
```

```
C:\Users\LENOVO\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWa
rning:

Function plot_confusion_matrix is deprecated; Function `plot_confusion_matrix` is dep
recated in 1.0 and will be removed in 1.2. Use one of the class methods: ConfusionMat
rixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
```

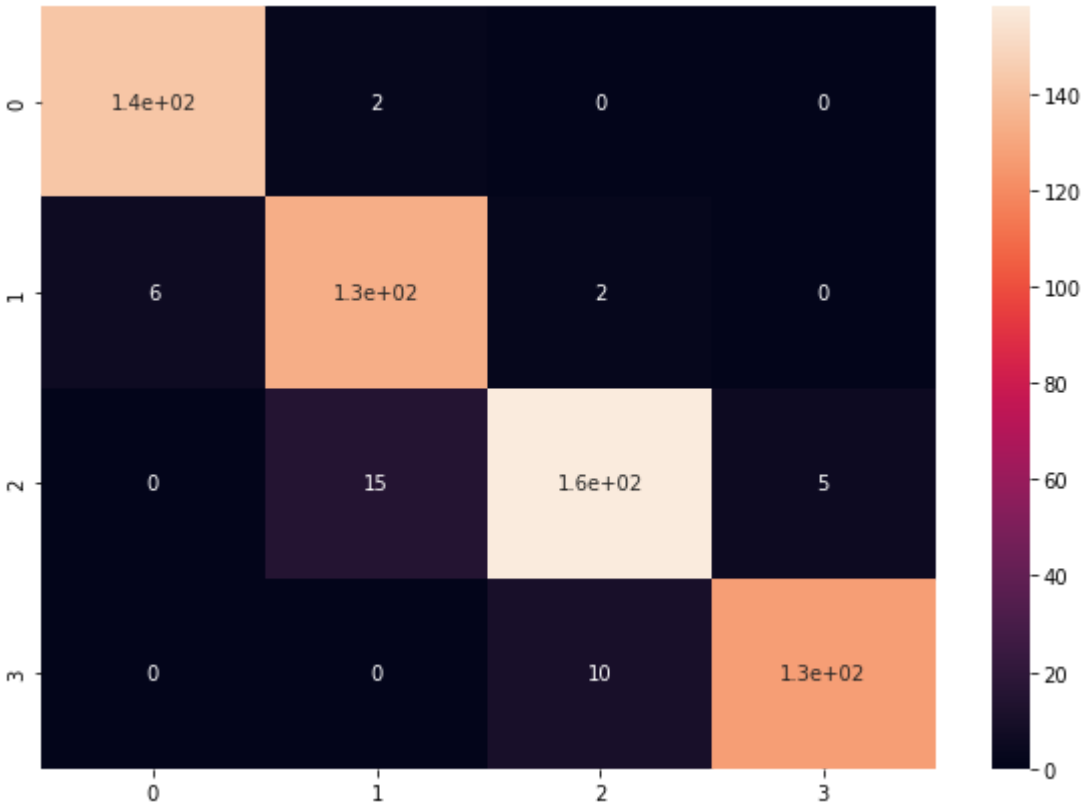
Out[84]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x14ee8203fd0>
```



In [85]:
```
#Plotting the confusion matrix for KNN
from sklearn.metrics import classification_report,confusion_matrix
prediction = knnc.predict(X_testData)
print(classification_report(y_testData,prediction))
matrix_val=confusion_matrix(y_testData,prediction)
print(matrix_val)
plot.figure(figsize = (10,7))
sbn.heatmap(matrix_val,annot=True)
```

```
                precision    recall  f1-score   support

           0       0.96      0.99      0.97       144
           1       0.89      0.94      0.91       141
           2       0.93      0.89      0.91       178
           3       0.96      0.93      0.94       137

    accuracy                           0.93       600
   macro avg       0.93      0.94      0.93       600
weighted avg       0.93      0.93      0.93       600

[[142   2   0   0]
 [  6 133   2   0]
 [  0  15 158   5]
 [  0   0  10 127]]
<AxesSubplot:>
```

Out[85]:



In [87]:

```
models = pds.DataFrame({
    'ML Algorithms': ['Logistic Regression', 'KNN'],
    'Accuracy %': [lr_accscore, knn_accscore]
})

models.sort_values(by = 'Accuracy %', ascending = False)
```

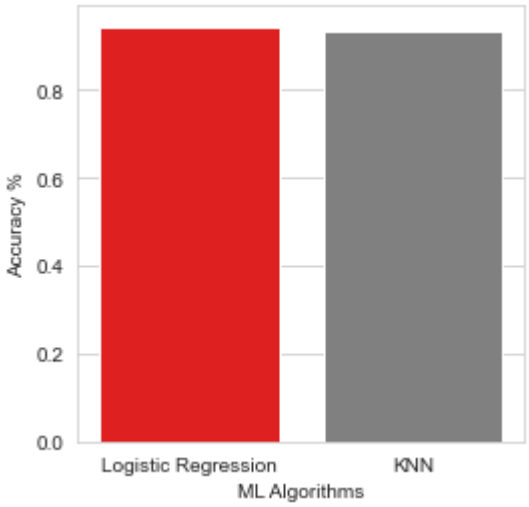
Out[87]:

	ML Algorithms	Accuracy %
0	Logistic Regression	0.945000
1	KNN	0.933333

In [90]:

```
#Plotting the barchart for accuracies from Logistic regression and KNN
colors = ["red", "gray"]
```

```
sbn.set_style("whitegrid")
plot.figure(figsize=(4,4))
sbn.barplot(x=models['ML Algorithms'],y=models['Accuracy %'], palette=colors )
plot.show()
```



Cross validating the model output.

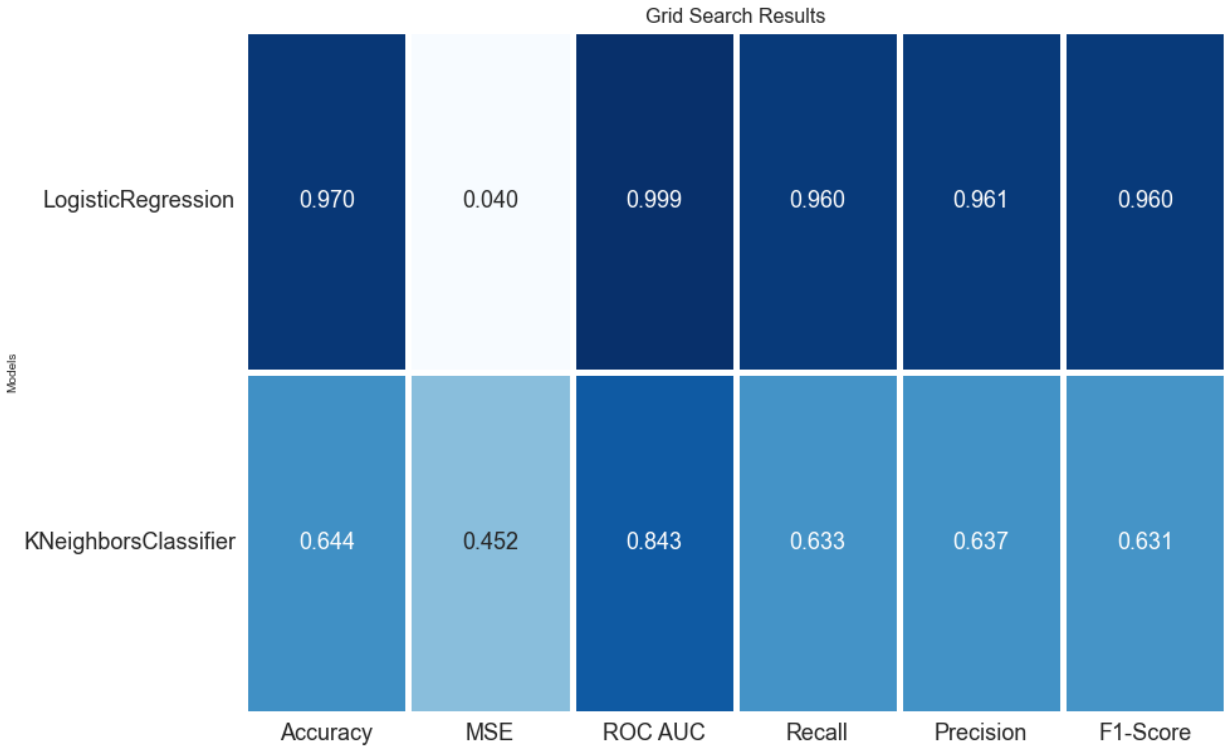
In [96]:
```
#Performing cross validation from obtained model results.

from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import GridSearchCV

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score
from sklearn.metrics import auc
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import f1_score
from sklearn.metrics import roc_curve
random_state = 101
MLclassifier = [LogisticRegression(random_state = random_state),
             KNeighborsClassifier()]

lr_grid = {"C":npy.logspace(-4, 4, 20),
                   "penalty": ["l1","l2","none"]}

knn_grid = {"n_neighbors": npy.linspace(2,20,12, dtype = int).tolist(),
                "weights": ["uniform","distance"],
                "metric":["euclidean","manhattan","minkowski"],
                "leaf_size": [1,3,5,12,30]}

MLclassifier_param = [lr_grid,
                  knn_grid]

crossval_res = []
best_est = []
mse_errors = []
roc_scores = []
```

```
recall_scores = []
precision = []
f1_scores = []

for a in range(len(MLclassifier)):
    print("========================================================================")
    clsfrs = GridSearchCV(MLclassifier[a],
                          param_grid=MLclassifier_param[a],
                          cv = StratifiedKFold(n_splits = 10),
                          scoring = "accuracy",
                          n_jobs = -1,verbose = 2)

    clsfrs.fit(X_train_s,y_trainData)

    crossval_res.append(clsfrs.best_score_)

    mse_errors.append(mean_squared_error(y_testData,clsfrs.predict(X_test_s)))

    roc_scores.append(roc_auc_score(y_testData, clsfrs.predict_proba(X_test_s), multi_

    recall_scores.append(recall_score(y_testData, clsfrs.predict(X_test_s), average='w

    precision.append(precision_score(y_testData, clsfrs.predict(X_test_s), average='we

    f1_scores.append(f1_score(y_testData, clsfrs.predict(X_test_s), average='weighted'

    best_est.append(clsfrs.best_estimator_)

    print("Model: {}".format(MLclassifier[a]))
    print("Accuracy: %{}".format(round(crossval_res[a]*100,2)))
    print("MSE: {}".format(mse_errors[a]))
    print("ROC AUC: {}".format(roc_scores[a]))
    print("Recall: {}".format(recall_scores[a]))
    print("Precision: {}".format(precision[a]))
    print("F1-Score: {}".format(f1_scores[a]))
    print("Best Estimator: {}".format(clsfrs.best_estimator_))

print("========================================================================")

sbn.set_style("darkgrid")
crossval_results = pds.DataFrame({"Accuracy":crossval_res,
                          "MSE":mse_errors,
                          "ROC AUC":roc_scores,
                          "Recall": recall_scores,
                          "Precision": precision,
                          "F1-Score":f1_scores,
                          "Models":["LogisticRegression",
                                    "KNeighborsClassifier"]})

crossval_results.index = crossval_results["Models"]

crossval_results  = crossval_results.drop(["Models"], axis = 1)

f,ax = plot.subplots(figsize=(14,10))

sbn.heatmap(crossval_results, annot=True,cmap = "Blues",fmt= '.3f',
            ax=ax,linewidths = 5, cbar = False,
            annot_kws={"size": 18})

plot.xticks(size = 18)
```

```
plot.yticks(size = 18, rotation = 0)
plot.ylabel("Models")
plot.title("Grid Search Results", size = 16)
plot.show()
```

```
================================================================================
Fitting 10 folds for each of 60 candidates, totalling 600 fits
C:\Users\LENOVO\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:372: FitFailedWarning:


200 fits failed out of a total of 600.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
200 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\LENOVO\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\LENOVO\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 1461, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\LENOVO\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 447, in _check_solver
    raise ValueError(
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.


C:\Users\LENOVO\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:969: UserWarning:

One or more of the test scores are non-finite: [       nan 0.58214286 0.96071429
nan 0.58928571 0.96071429
        nan 0.60285714 0.96071429        nan 0.64142857 0.96071429
        nan 0.71       0.96071429        nan 0.78357143 0.96071429
        nan 0.85785714 0.96071429        nan 0.91357143 0.96071429
        nan 0.94428571 0.96071429        nan 0.95857143 0.96071429
        nan 0.96571429 0.96071429        nan 0.96857143 0.96071429
        nan 0.96928571 0.96071429        nan 0.97       0.96071429
        nan 0.96785714 0.96071429        nan 0.965      0.96071429
        nan 0.96642857 0.96071429        nan 0.96428571 0.96071429
        nan 0.965      0.96071429        nan 0.96214286 0.96071429]

C:\Users\LENOVO\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning:

lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
```

```
Model: LogisticRegression(random_state=101)
Accuracy: %97.0
MSE: 0.04
ROC AUC: 0.9986023478673542
Recall: 0.96
Precision: 0.9607577849574407
F1-Score: 0.9598402316352899
Best Estimator: LogisticRegression(C=29.763514416313132, random_state=101)
========================================================================
Fitting 10 folds for each of 360 candidates, totalling 3600 fits
Model: KNeighborsClassifier()
Accuracy: %64.43
MSE: 0.45166666666666666
ROC AUC: 0.8434479103214669
Recall: 0.6333333333333333
Precision: 0.6370557028528684
F1-Score: 0.6313286163235683
Best Estimator: KNeighborsClassifier(leaf_size=1, metric='manhattan', n_neighbors=20,
                     weights='distance')
========================================================================
```



In [99]:
```
#Plotting graph for cross validation scores.
sbn.set_style("darkgrid")
crossval_results = pds.DataFrame({"Cross Validation Means":crossval_res,
                          "Models":["LogisticRegression",
                                    "KNeighborsClassifier"]})

plot.figure(figsize = (10,6))
sbn.barplot("Cross Validation Means", "Models",
            data = crossval_results, palette = "Set1")
plot.xlabel("Mean Accuracy",
           size = 12)
plot.yticks(size = 14)
plot.title("Cross Validation Scores",
          size = 12)
plot.show()
```

```
C:\Users\LENOVO\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:

Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
```