Question 5

(a)

LDAs work best on test data because the model may overfit the linearity on the Bayes decision boundary. QDAs work best on training data because the model is highly flexible which yields a closer t.

(b)

QDAs work best on both training and test sets when the Bayes decision boundary is nonlinear. It doesn't matter whether test or training data is being used because each class will have its own covariance matrix. LDAs have linear bounds with lower variance while QDAs have nonlinear bounds and higher variance.

(c)

As n increases, the test prediction accuracy of QDA relative to LDA will increase because the significance of variance (error) decreases.

(d)

This is false because the QDA will overfit the data given its high variance and the flexibility of the model.

Question 7

(a)

$\widehat{X}_Y=10, \widehat{X}_N=0$

$ \hat{\sigma^2}=36 $

$ P(Y)=.8, P(N)=.2

$$P(X=4 \vee Y)=\frac{1}{\sigma\sqrt{2\pi}}e^{\frac{-(X-\hat{X})^2}{2\sigma^2}} ¿ \frac{1}{6\sqrt{2\pi}}e^{\frac{-(4-10)^2}{2*36}}=0.0403$$

$$P(X=4 \vee N)=\frac{1}{\sigma\sqrt{2\pi}}e^{\frac{-(X)^2}{2\sigma^2}} ¿ \frac{1}{6\sqrt{2\pi}}e^{\frac{-(4)^2}{2*36}}=0.0532$$

$$P(X=4)=P(X=4 \vee Y)P(Y)+P(X=4 \vee N)P(N) ¿ 0.0403*0.8+0.0532*.2=0.0429$$

$P(Y \vee X=4)=\frac{P(Y)P(X=4 \vee Y)}{P(X=4)}$ \$=\frac{.0403.*8}{0.0429}=.7515*100% =75.2% \$

Question 16

```
from dataclasses import dataclass
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import mean_squared_error
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

import matplotlib.pyplot as plt

import seaborn as sns
Boston = pd.read_csv('Boston.csv')
Boston.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  506 non-null    int64
 1   crim        506 non-null    float64
 2   zn          506 non-null    float64
 3   indus       506 non-null    float64
 4   chas        506 non-null    int64
 5   nox         506 non-null    float64
 6   rm          506 non-null    float64
 7   age         506 non-null    float64
 8   dis         506 non-null    float64
 9   rad         506 non-null    int64
 10  tax         506 non-null    int64
 11  ptratio     506 non-null    float64
 12  lstat       506 non-null    float64
 13  medv        506 non-null    float64
dtypes: float64(10), int64(4)
memory usage: 55.5 KB
```

```
Boston.head()
```

```
   Unnamed: 0     crim    zn  indus  chas    nox     rm   age     dis
rad  \
0           1  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900
```

```
1
1           2  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671
2
2           3  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671
2
3           4  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622
3
4           5  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622
3

   tax  ptratio  lstat  medv
0  296     15.3   4.98  24.0
1  242     17.8   9.14  21.6
2  242     17.8   4.03  34.7
3  222     18.7   2.94  33.4
4  222     18.7   5.33  36.2
```

Predict whether a given suburb has a crime rate above or below the median.

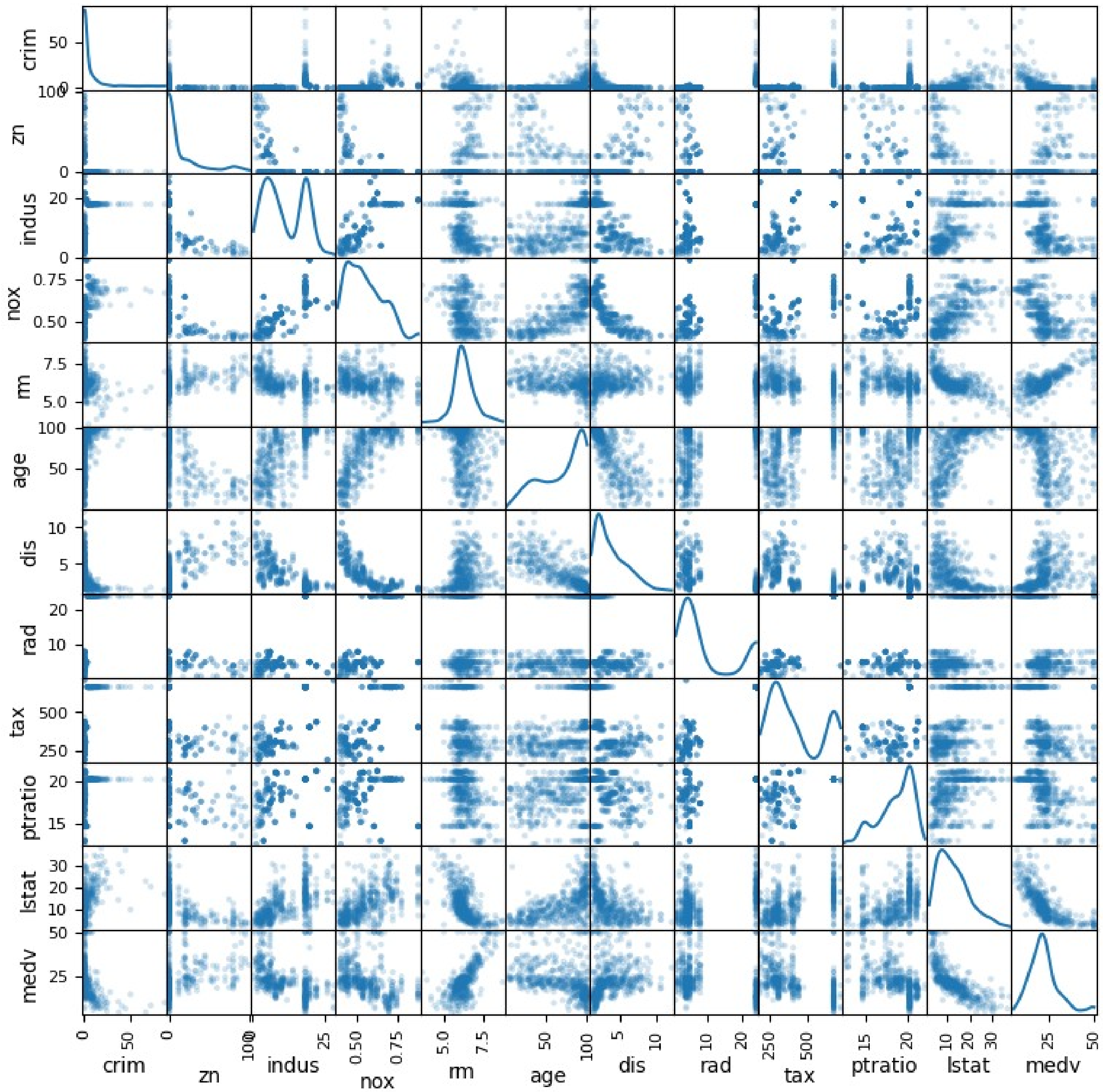
- Logistic regression
- LDA
- Naive Bayes
- KNN

```
Boston.drop(columns=Boston.columns[0], axis=1, inplace=True)
df = pd.DataFrame(data=Boston)
pd.plotting.scatter_matrix(df.drop('chas', axis=1, inplace=False),
alpha=0.2, figsize=(9,9), diagonal='kde')

array([[<Axes: xlabel='crim', ylabel='crim'>,
        <Axes: xlabel='zn', ylabel='crim'>,
        <Axes: xlabel='indus', ylabel='crim'>,
        <Axes: xlabel='nox', ylabel='crim'>,
        <Axes: xlabel='rm', ylabel='crim'>,
        <Axes: xlabel='age', ylabel='crim'>,
        <Axes: xlabel='dis', ylabel='crim'>,
        <Axes: xlabel='rad', ylabel='crim'>,
        <Axes: xlabel='tax', ylabel='crim'>,
        <Axes: xlabel='ptratio', ylabel='crim'>,
        <Axes: xlabel='lstat', ylabel='crim'>,
        <Axes: xlabel='medv', ylabel='crim'>],
       [<Axes: xlabel='crim', ylabel='zn'>,
        <Axes: xlabel='zn', ylabel='zn'>,
        <Axes: xlabel='indus', ylabel='zn'>,
        <Axes: xlabel='nox', ylabel='zn'>,
        <Axes: xlabel='rm', ylabel='zn'>,
        <Axes: xlabel='age', ylabel='zn'>,
        <Axes: xlabel='dis', ylabel='zn'>,
        <Axes: xlabel='rad', ylabel='zn'>,
        <Axes: xlabel='tax', ylabel='zn'>,
```

```
 <Axes: xlabel='ptratio', ylabel='zn'>,
 <Axes: xlabel='lstat', ylabel='zn'>,
 <Axes: xlabel='medv', ylabel='zn'>],
[<Axes: xlabel='crim', ylabel='indus'>,
 <Axes: xlabel='zn', ylabel='indus'>,
 <Axes: xlabel='indus', ylabel='indus'>,
 <Axes: xlabel='nox', ylabel='indus'>,
 <Axes: xlabel='rm', ylabel='indus'>,
 <Axes: xlabel='age', ylabel='indus'>,
 <Axes: xlabel='dis', ylabel='indus'>,
 <Axes: xlabel='rad', ylabel='indus'>,
 <Axes: xlabel='tax', ylabel='indus'>,
 <Axes: xlabel='ptratio', ylabel='indus'>,
 <Axes: xlabel='lstat', ylabel='indus'>,
 <Axes: xlabel='medv', ylabel='indus'>],
[<Axes: xlabel='crim', ylabel='nox'>,
 <Axes: xlabel='zn', ylabel='nox'>,
 <Axes: xlabel='indus', ylabel='nox'>,
 <Axes: xlabel='nox', ylabel='nox'>,
 <Axes: xlabel='rm', ylabel='nox'>,
 <Axes: xlabel='age', ylabel='nox'>,
 <Axes: xlabel='dis', ylabel='nox'>,
 <Axes: xlabel='rad', ylabel='nox'>,
 <Axes: xlabel='tax', ylabel='nox'>,
 <Axes: xlabel='ptratio', ylabel='nox'>,
 <Axes: xlabel='lstat', ylabel='nox'>,
 <Axes: xlabel='medv', ylabel='nox'>],
[<Axes: xlabel='crim', ylabel='rm'>,
 <Axes: xlabel='zn', ylabel='rm'>,
 <Axes: xlabel='indus', ylabel='rm'>,
 <Axes: xlabel='nox', ylabel='rm'>,
 <Axes: xlabel='rm', ylabel='rm'>,
 <Axes: xlabel='age', ylabel='rm'>,
 <Axes: xlabel='dis', ylabel='rm'>,
 <Axes: xlabel='rad', ylabel='rm'>,
 <Axes: xlabel='tax', ylabel='rm'>,
 <Axes: xlabel='ptratio', ylabel='rm'>,
 <Axes: xlabel='lstat', ylabel='rm'>,
 <Axes: xlabel='medv', ylabel='rm'>],
[<Axes: xlabel='crim', ylabel='age'>,
 <Axes: xlabel='zn', ylabel='age'>,
 <Axes: xlabel='indus', ylabel='age'>,
 <Axes: xlabel='nox', ylabel='age'>,
 <Axes: xlabel='rm', ylabel='age'>,
 <Axes: xlabel='age', ylabel='age'>,
 <Axes: xlabel='dis', ylabel='age'>,
 <Axes: xlabel='rad', ylabel='age'>,
 <Axes: xlabel='tax', ylabel='age'>,
 <Axes: xlabel='ptratio', ylabel='age'>,
```

```
         <Axes: xlabel='lstat', ylabel='age'>,
         <Axes: xlabel='medv', ylabel='age'>],
        [<Axes: xlabel='crim', ylabel='dis'>,
         <Axes: xlabel='zn', ylabel='dis'>,
         <Axes: xlabel='indus', ylabel='dis'>,
         <Axes: xlabel='nox', ylabel='dis'>,
         <Axes: xlabel='rm', ylabel='dis'>,
         <Axes: xlabel='age', ylabel='dis'>,
         <Axes: xlabel='dis', ylabel='dis'>,
         <Axes: xlabel='rad', ylabel='dis'>,
         <Axes: xlabel='tax', ylabel='dis'>,
         <Axes: xlabel='ptratio', ylabel='dis'>,
         <Axes: xlabel='lstat', ylabel='dis'>,
         <Axes: xlabel='medv', ylabel='dis'>],
        [<Axes: xlabel='crim', ylabel='rad'>,
         <Axes: xlabel='zn', ylabel='rad'>,
         <Axes: xlabel='indus', ylabel='rad'>,
         <Axes: xlabel='nox', ylabel='rad'>,
         <Axes: xlabel='rm', ylabel='rad'>,
         <Axes: xlabel='age', ylabel='rad'>,
         <Axes: xlabel='dis', ylabel='rad'>,
         <Axes: xlabel='rad', ylabel='rad'>,
         <Axes: xlabel='tax', ylabel='rad'>,
         <Axes: xlabel='ptratio', ylabel='rad'>,
         <Axes: xlabel='lstat', ylabel='rad'>,
         <Axes: xlabel='medv', ylabel='rad'>],
        [<Axes: xlabel='crim', ylabel='tax'>,
         <Axes: xlabel='zn', ylabel='tax'>,
         <Axes: xlabel='indus', ylabel='tax'>,
         <Axes: xlabel='nox', ylabel='tax'>,
         <Axes: xlabel='rm', ylabel='tax'>,
         <Axes: xlabel='age', ylabel='tax'>,
         <Axes: xlabel='dis', ylabel='tax'>,
         <Axes: xlabel='rad', ylabel='tax'>,
         <Axes: xlabel='tax', ylabel='tax'>,
         <Axes: xlabel='ptratio', ylabel='tax'>,
         <Axes: xlabel='lstat', ylabel='tax'>,
         <Axes: xlabel='medv', ylabel='tax'>],
        [<Axes: xlabel='crim', ylabel='ptratio'>,
         <Axes: xlabel='zn', ylabel='ptratio'>,
         <Axes: xlabel='indus', ylabel='ptratio'>,
         <Axes: xlabel='nox', ylabel='ptratio'>,
         <Axes: xlabel='rm', ylabel='ptratio'>,
         <Axes: xlabel='age', ylabel='ptratio'>,
         <Axes: xlabel='dis', ylabel='ptratio'>,
         <Axes: xlabel='rad', ylabel='ptratio'>,
         <Axes: xlabel='tax', ylabel='ptratio'>,
         <Axes: xlabel='ptratio', ylabel='ptratio'>,
         <Axes: xlabel='lstat', ylabel='ptratio'>,
```

```
        <Axes: xlabel='medv', ylabel='ptratio'>],
       [<Axes: xlabel='crim', ylabel='lstat'>,
        <Axes: xlabel='zn', ylabel='lstat'>,
        <Axes: xlabel='indus', ylabel='lstat'>,
        <Axes: xlabel='nox', ylabel='lstat'>,
        <Axes: xlabel='rm', ylabel='lstat'>,
        <Axes: xlabel='age', ylabel='lstat'>,
        <Axes: xlabel='dis', ylabel='lstat'>,
        <Axes: xlabel='rad', ylabel='lstat'>,
        <Axes: xlabel='tax', ylabel='lstat'>,
        <Axes: xlabel='ptratio', ylabel='lstat'>,
        <Axes: xlabel='lstat', ylabel='lstat'>,
        <Axes: xlabel='medv', ylabel='lstat'>],
       [<Axes: xlabel='crim', ylabel='medv'>,
        <Axes: xlabel='zn', ylabel='medv'>,
        <Axes: xlabel='indus', ylabel='medv'>,
        <Axes: xlabel='nox', ylabel='medv'>,
        <Axes: xlabel='rm', ylabel='medv'>,
        <Axes: xlabel='age', ylabel='medv'>,
        <Axes: xlabel='dis', ylabel='medv'>,
        <Axes: xlabel='rad', ylabel='medv'>,
        <Axes: xlabel='tax', ylabel='medv'>,
        <Axes: xlabel='ptratio', ylabel='medv'>,
        <Axes: xlabel='lstat', ylabel='medv'>,
        <Axes: xlabel='medv', ylabel='medv'>]], dtype=object)
```




```python
# (df[['chas']] == 0).sum()
df[['chas']].info()
# chas caterigorical -> 35 nonzero val
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   chas    506 non-null    int64
dtypes: int64(1)
memory usage: 4.1 KB
```

```python
# Convert crim to binary category
## 1 if crime rate is above the median
## 0 if crime rate is below the median

df['crim'] = np.where(df['crim'] >= df['crim'].median(), 1, 0)
((df['crim'] == 0).sum() == (df['crim'] == 1).sum() and (df['crim'] ==
0).sum() != 0)
```

```
np.True_
```

```python
@dataclass
class T:
    train: pd.DataFrame
    test: pd.DataFrame

"""
@TODO expand T to
    X_train, y_train,
    X_test, y_test
    mean_squared_error (MSE)
    prediction
    cross_validation
    references given a classification model (LDA, naive Bayes, KNN
models)
"""
def sample(df):
    sample = df.sample(frac=1, random_state=42).reset_index(drop=True)
    size = int(len(sample) * .8)
    train = sample[:size] # 80%
    test = sample[size:] # 20%

    return T(train=train, test=test)

data = sample(df)
data
```

```
T(train=     crim    zn  indus  chas    nox     rm   age     dis  rad
tax  ptratio  \
0       0   0.0   4.05     0  0.510  6.416  84.1  2.6463    5  296
16.6
1       0  40.0   6.41     1  0.447  6.758  32.9  4.0776    4  254
17.6
2       0   0.0  27.74     0  0.609  5.983  98.8  1.8681    4  711
20.1
3       0   0.0  10.81     0  0.413  6.065   7.8  5.2873    4  305
19.2
4       1   0.0  18.10     0  0.713  6.297  91.8  2.3682   24  666
20.2
..    ...   ...    ...   ...    ...    ...   ...     ...  ...  ...
...
```

```
399      1   0.0  18.10     0  0.740  5.627  93.9  1.8172   24  666
20.2
400      0   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273
21.0
401      0   0.0   5.19     0  0.515  5.895  59.6  5.6150    5  224
20.2
402      1   0.0   6.20     1  0.507  6.631  76.5  4.1480    8  307
17.4
403      0   0.0  10.59     0  0.489  5.783  72.7  4.3549    4  277
18.6

     lstat  medv
0     9.04  23.6
1     3.53  32.4
2    18.07  13.6
3     5.52  22.8
4    17.27  16.1
..     ...   ...
399  22.88  12.8
400   6.48  22.0
401  10.56  18.5
402   9.54  25.1
403  18.06  22.5

[404 rows x 13 columns], test=     crim    zn  indus  chas    nox
rm   age     dis  rad  tax  ptratio  \
404      0   0.0  10.59     1  0.489  5.807  53.8  3.6526    4  277
18.6
405      0   0.0  13.92     0  0.437  6.678  31.1  5.9604    4  289
16.0
406      1   0.0  18.10     0  0.713  6.317  83.0  2.7344   24  666
20.2
407      0  22.0   5.86     0  0.431  6.438   8.9  7.3967    7  330
19.1
408      1   0.0   9.69     0  0.585  5.926  42.6  2.3817    6  391
19.2
..     ...   ...    ...   ...    ...    ...   ...     ...  ...  ...
...
501      0   0.0   8.56     0  0.520  5.836  91.9  2.2110    5  384
20.9
502      1  20.0   6.96     0  0.464  5.856  42.1  4.4290    3  223
18.6
503      0  80.0   2.01     0  0.435  6.635  29.7  8.3440    4  280
17.0
504      1   0.0  18.10     0  0.740  6.629  94.6  2.1247   24  666
20.2
505      0   0.0   8.56     0  0.520  6.405  85.4  2.7147    5  384
20.9

     lstat  medv
```

```
404  16.03  22.4
405   6.27  28.6
406  13.99  19.5
407   3.59  24.8
408  13.59  24.5
..     ...   ...
501  18.66  19.5
502  13.00  21.1
503   5.99  24.5
504  23.27  13.4
505  10.63  18.6

[102 rows x 13 columns])
```

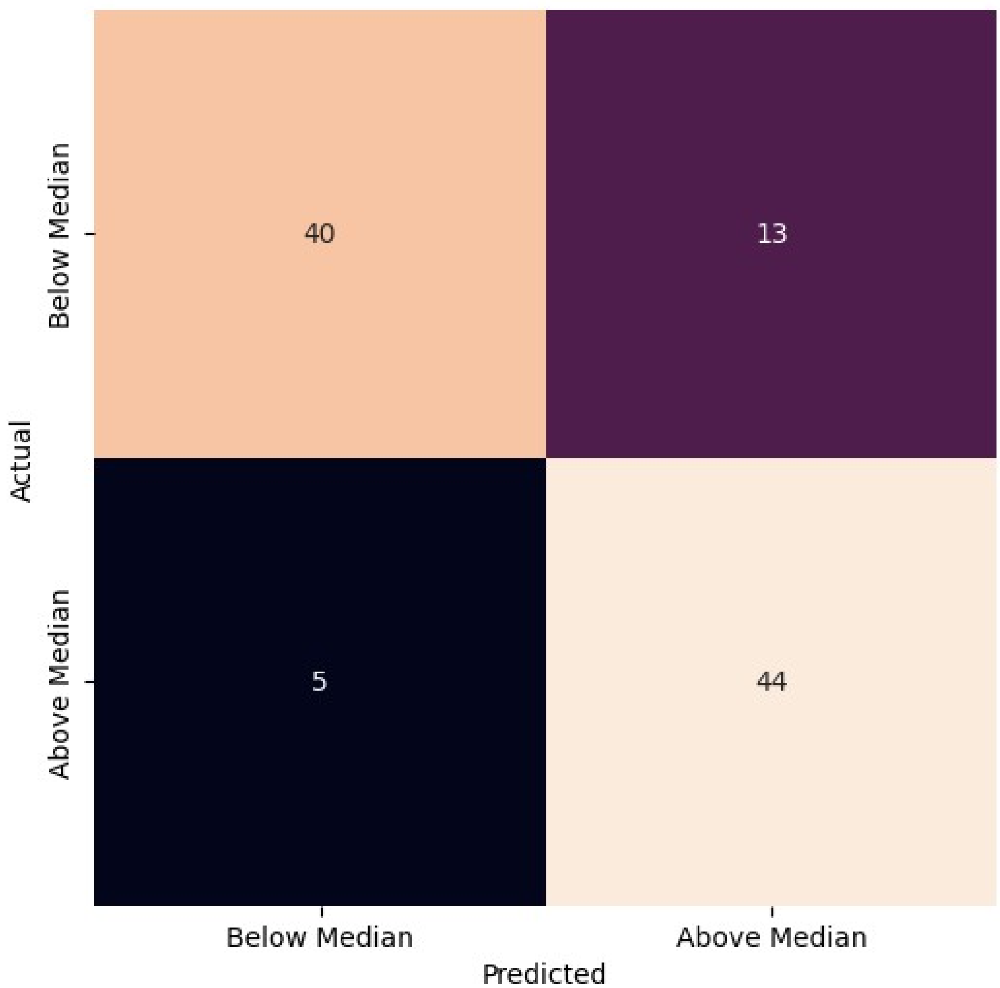
```
X_train = data.train[['zn', 'chas', 'nox', 'rm', 'age', 'dis',
'lstat', 'medv']]
y_train = data.train[['crim']]
X_test = data.test[['zn', 'chas', 'nox', 'rm', 'age', 'dis', 'lstat',
'medv']]
y_test = data.test[['crim']]

def trainModel(model):
    model.fit(X_train, y_train.values.ravel())
    y_pred = model.predict(X_test)
    accuracy = (y_pred == y_test.values.ravel()).mean()
    print(accuracy)

def confusionMatrix(model):
    y_pred = model.predict(X_test)
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(6,6))
    sns.heatmap(cm, annot=True, fmt='d', cbar=False,
xticklabels=['Below Median', 'Above Median'],
            yticklabels=['Below Median', 'Above Median'])

    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()

from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score

def analysis(model):
    model.fit(X_train, y_train.values.ravel())
    y_pred = model.predict(X_test)

    print(classification_report(y_test, y_pred))
    scores = cross_val_score(model, X_train, y_train.values.ravel(),
cv=5, scoring='accuracy')
    print(f"Cross-Validation Scores: {scores}")
    print(f"Mean CV Accuracy: {scores.mean()}")
```

```
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(6,6))
    sns.heatmap(cm, annot=True, fmt='d', cbar=False,
xticklabels=['Below Median', 'Above Median'],
            yticklabels=['Below Median', 'Above Median'])

    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()
```

Logistic Regression

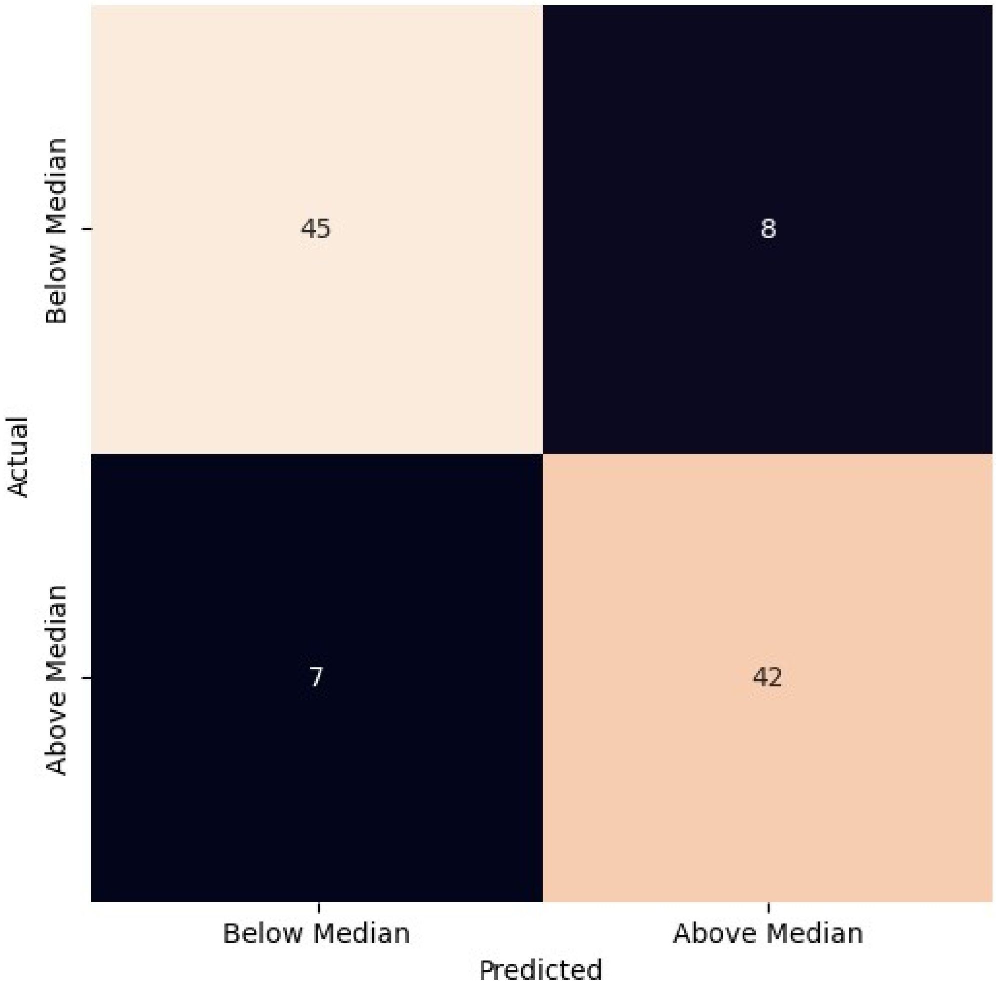
```
analysis(LogisticRegression(solver='lbfgs', max_iter=500))
```

```
              precision    recall  f1-score   support

           0       0.89      0.75      0.82        53
           1       0.77      0.90      0.83        49

    accuracy                           0.82       102
   macro avg       0.83      0.83      0.82       102
weighted avg       0.83      0.82      0.82       102

Cross-Validation Scores: [0.80246914 0.79012346 0.81481481 0.77777778
0.85      ]
Mean CV Accuracy: 0.807037037037037
```



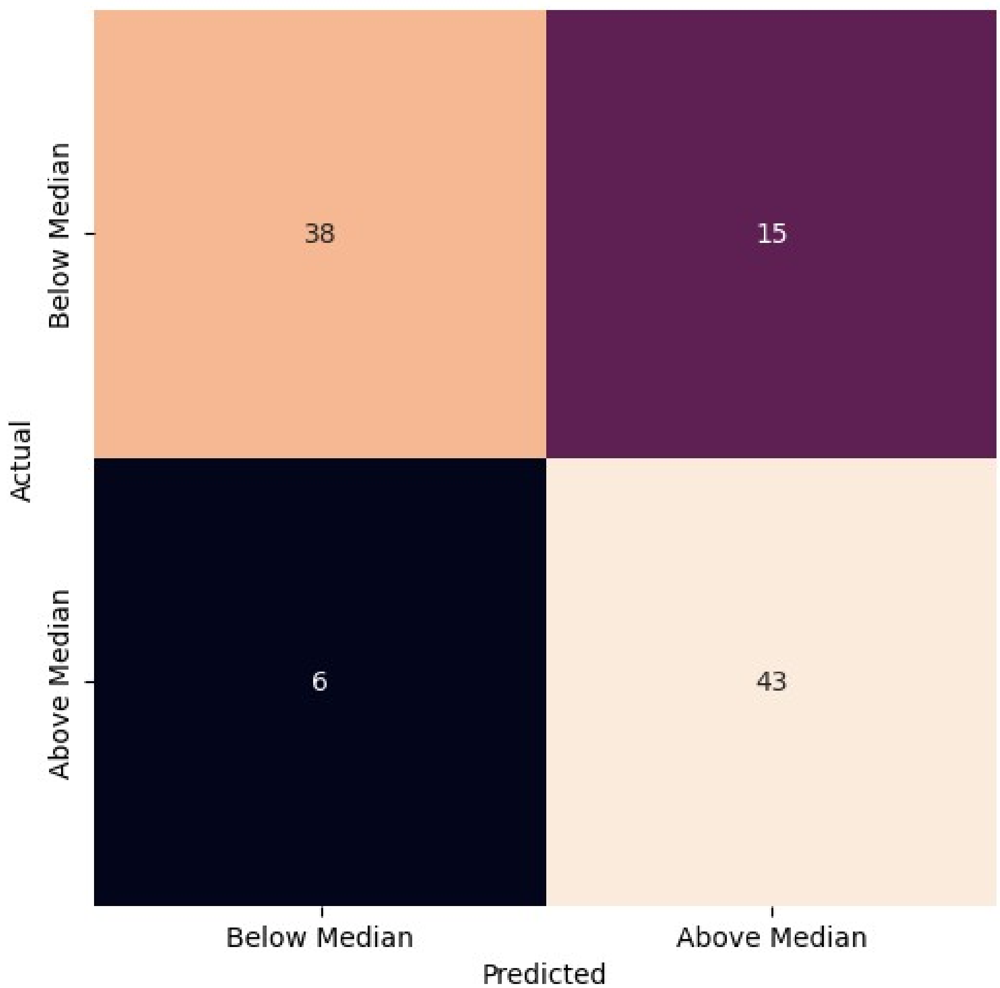
Linear Discriminant Analysis

```
analysis(LinearDiscriminantAnalysis())

              precision    recall  f1-score   support

           0       0.87      0.85      0.86        53
           1       0.84      0.86      0.85        49

    accuracy                           0.85       102
   macro avg       0.85      0.85      0.85       102
weighted avg       0.85      0.85      0.85       102

Cross-Validation Scores: [0.85185185 0.81481481 0.79012346 0.82716049
0.8625    ]
Mean CV Accuracy: 0.8292901234567902
```



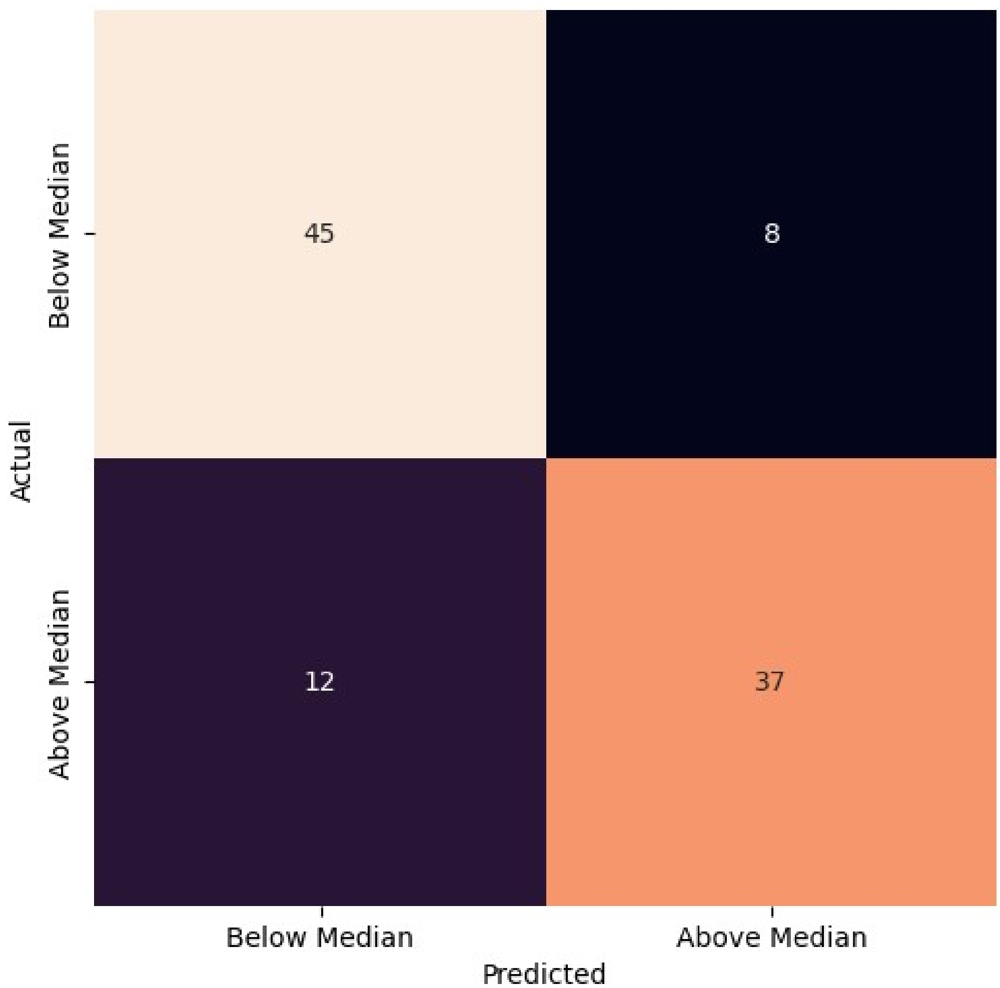
Naive Bayes

```
analysis(GaussianNB())

              precision    recall  f1-score   support

           0       0.86      0.72      0.78        53
           1       0.74      0.88      0.80        49

    accuracy                           0.79       102
   macro avg       0.80      0.80      0.79       102
weighted avg       0.80      0.79      0.79       102

Cross-Validation Scores: [0.80246914 0.85185185 0.74074074 0.81481481
0.825     ]
Mean CV Accuracy: 0.8069753086419753
```

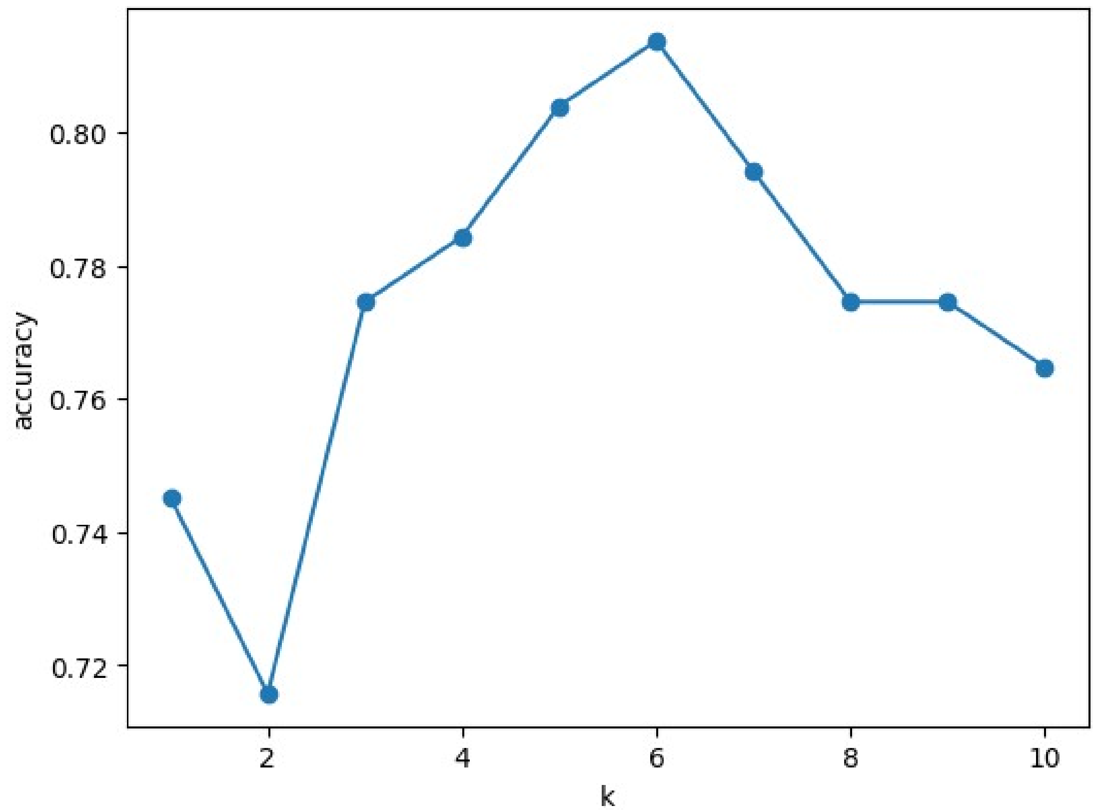


KNN

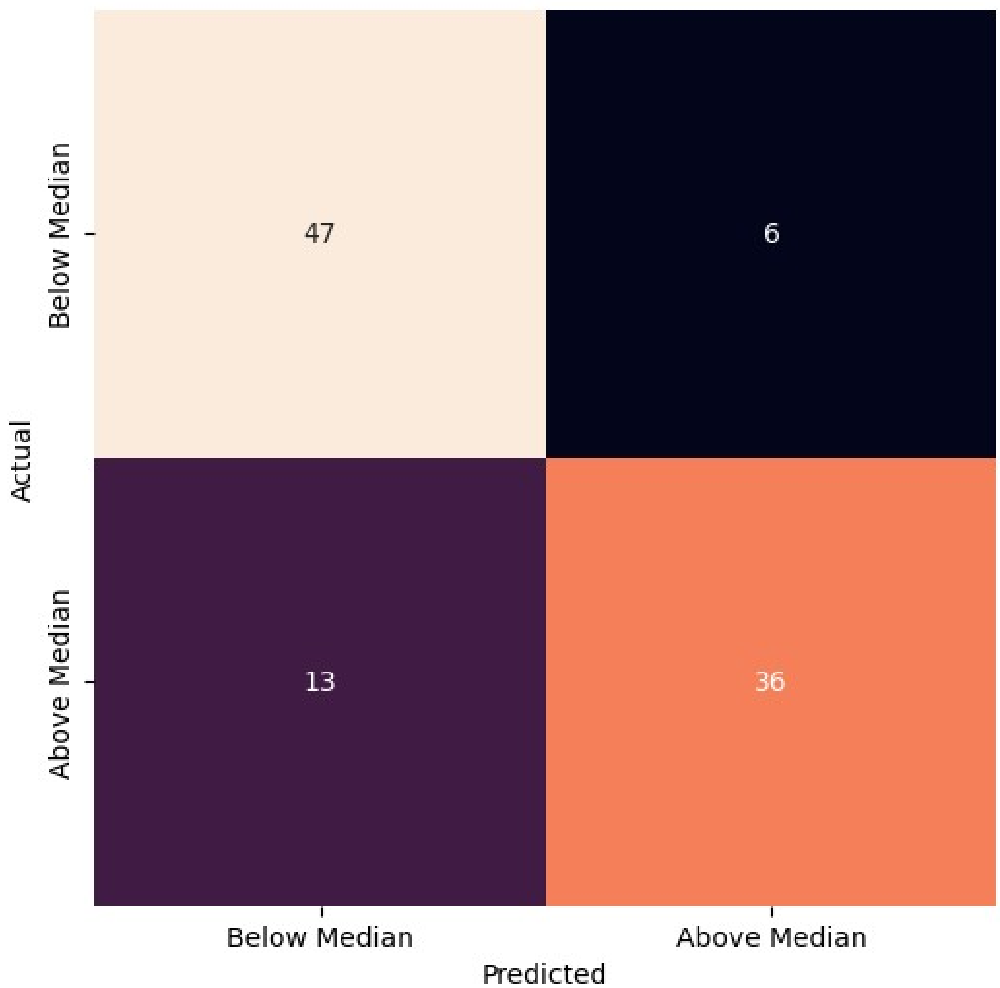
```
analysis(KNeighborsClassifier(n_neighbors=5))

              precision    recall  f1-score   support

           0       0.79      0.85      0.82        53
           1       0.82      0.76      0.79        49

    accuracy                           0.80       102
   macro avg       0.81      0.80      0.80       102
weighted avg       0.81      0.80      0.80       102

Cross-Validation Scores: [0.80246914 0.79012346 0.80246914 0.79012346
0.75      ]
Mean CV Accuracy: 0.787037037037037
```




```
accuracies = []
for k in range(1,11):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train.values.ravel())
    y_pred = knn.predict(X_test)

    accuracies.append(accuracy_score(y_test, y_pred))

plt.plot(range(1,11), accuracies, marker='o')
plt.xlabel('k')
plt.ylabel('accuracy')
plt.show()
```




```
analysis(KNeighborsClassifier(n_neighbors=6))

              precision    recall  f1-score   support

           0       0.78      0.89      0.83        53
           1       0.86      0.73      0.79        49

    accuracy                           0.81       102
   macro avg       0.82      0.81      0.81       102
weighted avg       0.82      0.81      0.81       102

Cross-Validation Scores: [0.80246914 0.80246914 0.79012346 0.7654321
0.7625    ]
Mean CV Accuracy: 0.7845987654320987
```



Overall, the Linear Discriminant Analysis model fits the data best with respect to cross validation and a classification report

Exercise 12

$a.¿\hat{\beta}_0+\hat{\beta}_1 x$

$b.¿\left(\hat{\alpha}_{orange,0}-\hat{\alpha}_{apple,0}x\right)+\left(\hat{\alpha}_{orange,1}-\hat{\alpha}_{apple,1}x\right)$

c.)

$\hat{\beta}_0=\hat{\alpha}_{orange,0}-\hat{\alpha}_{apple,0}x=2$

$\hat{\beta}_1=\hat{\alpha}_{orange,1}-\hat{\alpha}_{apple,1}x=-1$

d.)

$2=1.2-3x \rightarrow x=-0.27$

$$-1 = -2 + .6x \rightarrow x = 1.7$$