In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
import statistics
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.model_selection import LeaveOneOut
from sklearn.model_selection import KFold
```

In [2]:
```
df = pd.read_csv('College.csv')
df_copy = df.copy()

# Binarize 'Private' variable
mapping = {'Yes': 1.0, 'No': 0.0}
df_copy['Private'] = df_copy['Private'].replace(mapping)

# Drop the unnamed column
df_copy = df_copy.drop('Unnamed: 0', 1)

# Switch the order of columns
cols = df_copy.columns.tolist()
cols = cols[1:2] + cols[0:1] + cols[2:]
df_copy = df_copy[cols]
```

Problem a

In [3]:
```
y = np.asarray(df_copy['Apps'])
X = np.asarray(df_copy[cols[1:]])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.10)
```

Problem b

In [4]:
```
LinRegr_classifier = LinearRegression()
LinRegr_classifier.fit(X_train, y_train)

y_pred = LinRegr_classifier.predict(X_test)

# Calculate test error

RSS = np.sum((y_test - y_pred)**2)/len(y_test)
print('The testing error for the Linear Regression model is ' + repr(RSS))
```

```
The testing error for the Linear Regression model is 760696.3055473632
```

Problem c

In [5]:
```
# determine best value for lambda tuning parameter
possible_lambdas = np.linspace(0.01,10,2000)
best_lambda = 0
best_score = np.inf
K = 10
# Obtain test error rate for each value of Lambda
for value in (possible_lambdas):
    kfold = KFold(n_splits=K, shuffle=True)

    sum_test_errors = 0
    for train_index, test_index in kfold.split(X):
    #      print("TRAIN:", train_index, "TEST:", test_index)
        X_train_curr, X_test_curr = X[train_index], X[test_index]
        y_train_curr, y_test_curr = y[train_index], y[test_index]

        ridge_clf = Ridge(alpha=value)
        ridge_clf.fit(X_train_curr, y_train_curr)

        y_pred = ridge_clf.predict(X_test_curr)
        test_error = np.sum((y_test_curr - y_pred)**2)/len(y_test_curr)

        sum_test_errors = sum_test_errors + test_error

    current_test_error = sum_test_errors/K

    if current_test_error < best_score:
        best_score = current_test_error
        best_lambda = value
print("The best lambda value is " + repr(best_lambda))
```

```
The best lambda value is 0.5797148574287144
```

In [6]:
```
Ridge_classifier = Ridge(alpha=best_lambda)
Ridge_classifier.fit(X_train, y_train)

y_pred = ridge_clf.predict(X_test)
test_error = np.sum((y_test - y_pred)**2)/len(y_test)
print("The testing error for the Ridge Regression is " + repr(test_error))
```

```
The testing error for the Ridge Regression is 730401.5779168981
```

In [7]:
```
Ridge_classifier.intercept_
```

Out[7]:
```
-421.9170793817261
```

In [8]:
```
Ridge_classifier.coef_
```

Out[8]:
```
array([-4.70987558e+02,  1.60035171e+00, -9.42587922e-01,  5.28544620e+01,
       -1.58319009e+01,  6.40311736e-02,  5.31948484e-02, -8.35445544e-02,
        1.35880705e-01, -1.16938605e-02,  3.15748494e-02, -9.51811205e+00,
       -2.50291557e+00,  1.95851971e+01,  1.12261626e+00,  7.36896260e-02,
        8.37511736e+00])
```

In [ ]:
```
```

Problem d

In [9]:
```
# determine best value for lambda tuning parameter
possible_lambdas = np.linspace(0.01,10,2000)
best_lambda = 0
best_score = np.inf
K = 10
# Obtain test error rate for each value of Lambda
for value in (possible_lambdas):
    kfold = KFold(n_splits=K, shuffle=True)

    sum_test_errors = 0
    for train_index, test_index in kfold.split(X):
    #      print("TRAIN:", train_index, "TEST:", test_index)
        X_train_curr, X_test_curr = X[train_index], X[test_index]
        y_train_curr, y_test_curr = y[train_index], y[test_index]

        lasso_clf = Lasso(alpha=value)
        lasso_clf.fit(X_train_curr, y_train_curr)

        y_pred = lasso_clf.predict(X_test_curr)
        test_error = np.sum((y_test_curr - y_pred)**2)/len(y_test_curr)

        sum_test_errors = sum_test_errors + test_error

    current_test_error = sum_test_errors/K


    if current_test_error < best_score:
        best_score = current_test_error
        best_lambda = value
print("The best lambda value is " + repr(best_lambda))
```

```
The best lambda value is 7.076463231615808
```

In [10]:
```
Lasso_classifier = Lasso(alpha=best_lambda)
Lasso_classifier.fit(X_train, y_train)

y_pred = Lasso_classifier.predict(X_test)
test_error = np.sum((y_test - y_pred)**2)/len(y_test)
print("The testing error for the Lasso Regression is " + repr(test_error))
```

```
The testing error for the Lasso Regression is 762506.6969570713
```

In [11]:
```
Lasso_classifier.intercept_
```

Out[11]:
```
-498.85739784905354
```

In [12]:
```
Lasso_classifier.coef_
```

Out[12]:
```
array([-3.78954604e+02,  1.60067219e+00, -9.40639141e-01,  5.25415567e+01,
       -1.56528726e+01,  6.65948374e-02,  5.40575095e-02, -8.79094633e-02,
        1.32814380e-01, -1.55094990e-02,  3.08323857e-02, -9.12274596e+00,
       -2.06234017e+00,  2.01394966e+01,  7.60989075e-01,  7.42717745e-02,
        8.20597095e+00])
```

In [ ]:
```

```

In [ ]: