In [10]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
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.preprocessing import PolynomialFeatures
from sklearn.model_selection import KFold
```

In [11]:
```
df = pd.read_csv('Wage.csv')
df_copy = df.copy()
```

Problem a

In [12]:
```
# Extract predictor x and response y as arrays
y = np.asarray(df_copy['wage'])
x = np.reshape(np.asarray(df_copy['age']), (-1,1))


# Generate polynomial features up to degree 10
max_degree = 10
poly = PolynomialFeatures(degree=max_degree, include_bias=False)
x_new = poly.fit_transform(x)


K = 20
lowest_error = np.inf
best_degree = None
poss_degrees = np.linspace(1,max_degree,max_degree).astype(int)
for degree in (poss_degrees):
    # Obtain the terms of 1 to current degree
    x_curr = x_new[:, :degree]

    # K-Fold splitter
    kfold = KFold(n_splits=K, shuffle=True)
    sum_test_errors = 0 # Initialize the total test error
    for train_index, test_index in kfold.split(x_curr): # For each group
        # Obtain training and testing data
        X_train, X_test = x_curr[train_index], x_curr[test_index]
        y_train, y_test = y[train_index], y[test_index]

        # Create Linear Regression model and fit to the training data
        LinRegr_classifier = LinearRegression()
        LinRegr_classifier.fit(X_train, y_train)

        # Make predictions and calculate Residual Square Error
        y_pred = LinRegr_classifier.predict(X_test)
        test_error = np.sum((y_test - y_pred)**2)/len(y_test)

        sum_test_errors = sum_test_errors + test_error

    # cross-validated test error for polynomial model of degree "degree"
    current_test_error = sum_test_errors/K

    if current_test_error < lowest_error:
        lowest_error = current_test_error
```

```
        best_degree = degree

print("The best degree for polynomial model is " + repr(best_degree))
```

The best degree for polynomial model is 4

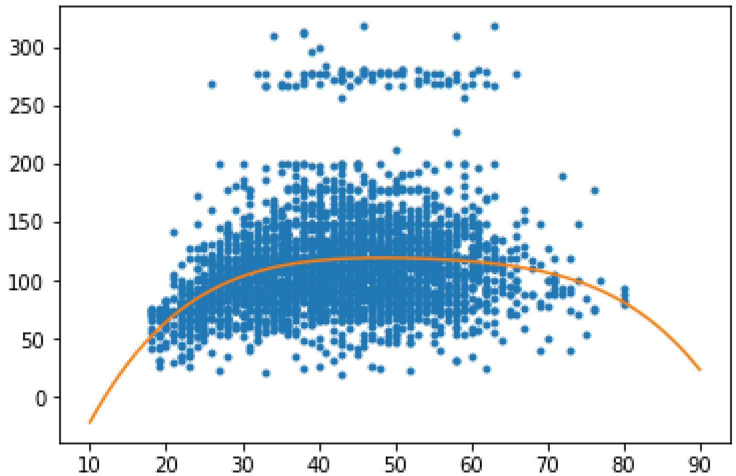
In [13]:

```
# Create Linear Regression and fit model of degree "best_degree"
LinRegr_classifier = LinearRegression()
LinRegr_classifier.fit(x_new[:, :best_degree], y)

# Create Linear Regression Function
Intercept = LinRegr_classifier.intercept_
Coefficients = LinRegr_classifier.coef_
x_values = np.linspace(10,90,3000)
function = 0
function = function + Intercept
for i in range(best_degree):
    function = function + Coefficients[i]*(x_values**(i+1))

plt.plot(x,y, '.')
plt.plot(x_values, function)
```

Out[13]: [<matplotlib.lines.Line2D at 0x7fe48a3c5550>]



Problem b

In [20]:
```
# Initialize important variables
best_num_cuts = None
lowest_score = np.inf
poss_cuts = np.linspace(1,20,20) # list of possible number of cuts
best_cut = None
best_bins = None

for cuts in (poss_cuts): # For each number of cuts ...
    # create dataframe with bins from number of cuts
    df_cut, bins = pd.cut(df['age'], bins=cuts, retbins = True, right = True)
    df_steps = pd.concat([df['age'], df_cut, df['wage']], keys=['age','age_cuts','wage'], axis = 1)

    # One-hot encode the age groups and extract the response 'wage'
    X = np.asarray(pd.get_dummies(df_steps['age_cuts']))
    X = sm.add_constant(X) # Statsmodels requires explicit adding of a constant (intercept)
    y = df_steps['wage']

    # Perform 10-fold cross validaton
    kfold = KFold(n_splits=10, shuffle=True)
    sum_test_errors = 0 # Initialize the total test error
    for train_index, test_index in kfold.split(X): # For each group
        # Obtain training and testing data
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]

        # fit the model
        step_fit= sm.GLM(y_train, X_train).fit()

        # make predictions and calculate test error
        y_pred = step_fit.predict(X_test)
        curr_error = np.sum((y_test - y_pred)**2)/(len(y_pred))

        # accumulate test errors
        sum_test_errors = sum_test_errors + curr_error

    # find average cross validation error
    avg_test_error = sum_test_errors/10

    # determine if current number of cuts is best
    if avg_test_error < lowest_score:
        lowest_score = avg_test_error
        best_num_cuts = cuts
```

```
        best_cut = X
        best_bins = bins

print(best_num_cuts)
```

```
16.0
```

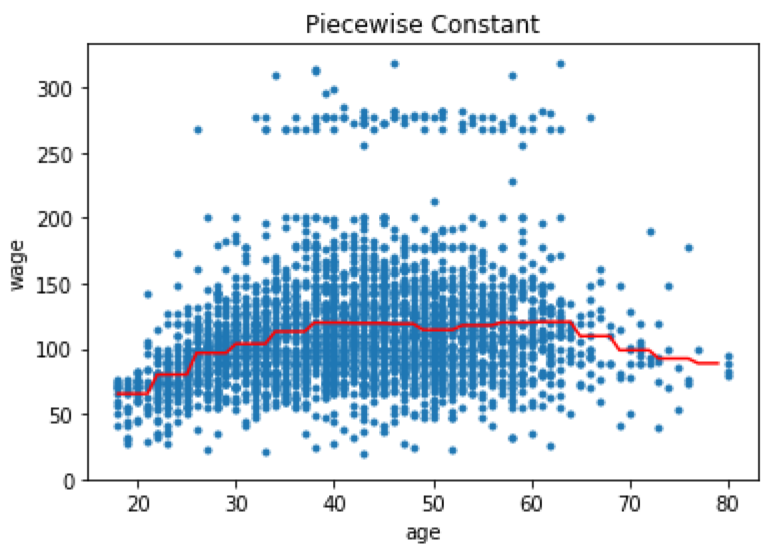
In [23]:
```
# Obtain step function model
step_fit = sm.GLM(df['wage'], best_cut).fit()

# Generate x and y data for graphing purposes
ages = np.arange(df['age'].min(), df['age'].max()).reshape(-1,1)
bin_mapping = np.digitize(ages.ravel(), best_bins)
data = sm.add_constant(pd.get_dummies(bin_mapping))

# Predict the value of the generated ages using the linear model
pred = step_fit.predict(data)


# Scatter plot with polynomial regression line
plt.plot(df['age'], df['wage'], '.')
plt.plot(ages, pred, c = 'r')
plt.title('Piecewise Constant')
plt.xlabel('age')
plt.ylabel('wage')
plt.ylim(ymin = 0)
```

Out[23]: (0, 333.25527471319606)



In [ ]:

In [ ]:

In [ ]:

In [ ]: