In [54]:
```
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.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
```

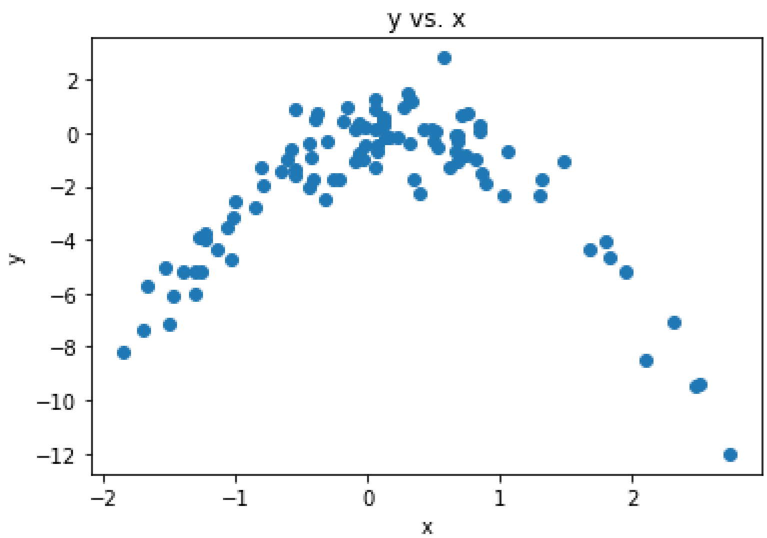
# Problem a

In [55]:
```
# normal distribution mean and standard deviation
mu = 0
sigma = 1

# X: 100 random samples from normal distribution N(mu = 0, std = 1)
x = np.random.normal(mu, sigma, 100)

# noise: 100 random samples from normal distribution N(mu = 0, std = 1)
noise = np.random.normal(mu, sigma, 100)

# Response: y = x - 2*x^2 + noise
y = x - 2*x**2 + noise
```

# Problem b

In [56]:
```
# Scatterplot of x and y
plt.scatter(x, y)
plt.title("y vs. x")
plt.xlabel("x")
plt.ylabel("y")
```

Out[56]: `Text(0,0.5,'y')`




# Problem c

# Model 1

In [67]:
```
# --------------------- MODEL 1 -----------------
# Make dataset of desired predictors
predictors = {
        'x': x,
       }
data = pd.DataFrame(predictors)
data = np.asarray(data)


# ----------------- LOOCV ---------------------
loo = LeaveOneOut()
n = loo.get_n_splits(data)
sum_MSE = 0
for train_index, test_index in loo.split(data):
    # Obtain training and testing data splits
    X_train, X_test = data[train_index], data[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Create inear Regression Model and fit data
    LinRegr_classifier = LinearRegression()
    LinRegr_classifier.fit(X_train, y_train)

    # Make prediction on test data
    y_pred = LinRegr_classifier.predict(X_test)

    # Calculate MSE
    current_mse = (y_test[0] - y_pred[0])**2

    sum_MSE = sum_MSE + current_mse

avg_mse = sum_MSE/n
print("The LOOCV error from Model 1 is " + repr(avg_mse))
```

```
The LOOCV error from Model 1 is 8.427931437291429
```

# Model 2

In [58]:
```
# --------------------- MODEL 2 -----------------
# Make dataset of desired predictors
predictors = {
        'x': x,
        'x^2': x**2,
       }
data = pd.DataFrame(predictors)
data = np.asarray(data)
data = sm.add_constant(data)


# ----------------- LOOCV ---------------------
loo = LeaveOneOut()
n = loo.get_n_splits(data)
sum_MSE = 0
for train_index, test_index in loo.split(data):
    # Obtain training and testing data splits
    X_train, X_test = data[train_index], data[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Create inear Regression Model and fit data
    LinRegr_classifier = LinearRegression()
    LinRegr_classifier.fit(X_train, y_train)

    # Make prediction on test data
    y_pred = LinRegr_classifier.predict(X_test)

    # Calculate MSE
    current_mse = (y_test[0] - y_pred[0])**2

    sum_MSE = sum_MSE + current_mse

avg_mse = sum_MSE/n
print("The LOOCV error from Model 2 is " + repr(avg_mse))
```

```
The LOOCV error from Model 2 is 0.882104115348292
```

# Model 3

In [59]:
```
# --------------------- MODEL 3 -----------------
# Make dataset of desired predictors
predictors = {
        'x': x,
        'x^2': x**2,
        'x^3': x**3,
       }
data = pd.DataFrame(predictors)
data = np.asarray(data)


# ----------------- LOOCV ---------------------
loo = LeaveOneOut()
n = loo.get_n_splits(data)
sum_MSE = 0
for train_index, test_index in loo.split(data):
    # Obtain training and testing data splits
    X_train, X_test = data[train_index], data[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Create inear Regression Model and fit data
    LinRegr_classifier = LinearRegression()
    LinRegr_classifier.fit(X_train, y_train)

    # Make prediction on test data
    y_pred = LinRegr_classifier.predict(X_test)

    # Calculate MSE
    current_mse = (y_test[0] - y_pred[0])**2

    sum_MSE = sum_MSE + current_mse

avg_mse = sum_MSE/n
print("The LOOCV error from Model 3 is " + repr(avg_mse))
```

```
The LOOCV error from Model 3 is 0.894180442253765
```

# Model 4

In [60]:
```
# --------------------- MODEL 4 -----------------
# Make dataset of desired predictors
predictors = {
        'x': x,
        'x^2': x**2,
        'x^3': x**3,
        'x^4': x**4,
       }
data = pd.DataFrame(predictors)
data = np.asarray(data)


# ----------------- LOOCV ---------------------
loo = LeaveOneOut()
n = loo.get_n_splits(data)
sum_MSE = 0
for train_index, test_index in loo.split(data):
    # Obtain training and testing data splits
    X_train, X_test = data[train_index], data[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Create inear Regression Model and fit data
    LinRegr_classifier = LinearRegression()
    LinRegr_classifier.fit(X_train, y_train)

    # Make prediction on test data
    y_pred = LinRegr_classifier.predict(X_test)

    # Calculate MSE
    current_mse = (y_test[0] - y_pred[0])**2

    sum_MSE = sum_MSE + current_mse

avg_mse = sum_MSE/n
print("The LOOCV error from Model 4 is " + repr(avg_mse))
```

```
The LOOCV error from Model 4 is 0.9063418943218408
```

# Problem d

In [68]:
```
# --------------------- MODEL 1 -----------------
# Make dataset of desired predictors
predictors = {
        'x': x,
       }
data = pd.DataFrame(predictors)
data = np.asarray(data)
data = sm.add_constant(data)


# Create the Multiple Linear Regression Model and fit it
MLRmodel = sm.OLS(y, data)
result = MLRmodel.fit()

# Create table of model prediction results
data = {'Coefficient Beta_i': result.params,
        't-Values': result.tvalues,
        'p-Values': result.pvalues
       }
data_analysis = pd.DataFrame(data)
data_analysis.round(4) # Round values in table to 4-decimal places
```

Out[68]:

| | Coefficient Beta_i | t-Values | p-Values |
|---|---|---|---|
| **0** | -1.8842 | -6.6805 | 0.0000 |
| **1** | -0.0785 | -0.2781 | 0.7815 |

In [70]:
```
# -------------------- MODEL 2 ----------------
# Make dataset of desired predictors
predictors = {
        'x': x,
        'x^2': x**2,
       }
data = pd.DataFrame(predictors)
data = np.asarray(data)
data = sm.add_constant(data)


# Create the Multiple Linear Regression Model and fit it
MLRmodel = sm.OLS(y, data)
result = MLRmodel.fit()

# Create table of model prediction results
data = {'Coefficient Beta_i': result.params,
        't-Values': result.tvalues,
        'p-Values': result.pvalues
       }
data_analysis = pd.DataFrame(data)
data_analysis.round(4) # Round values in table to 4-decimal places
```

Out[70]:

| | Coefficient Beta_i | t-Values | p-Values |
|---|---|---|---|
| **0** | -0.0511 | -0.4506 | 0.6533 |
| **1** | 1.0246 | 10.1348 | 0.0000 |
| **2** | -1.9310 | -28.3073 | 0.0000 |

In [71]:
```
# --------------------- MODEL 3 -----------------
# Make dataset of desired predictors
predictors = {
        'x': x,
        'x^2': x**2,
        'x^3': x**3,
       }
data = pd.DataFrame(predictors)
data = np.asarray(data)
data = sm.add_constant(data)


# Create the Multiple Linear Regression Model and fit it
MLRmodel = sm.OLS(y, data)
result = MLRmodel.fit()

# Create table of model prediction results
data = {'Coefficient Beta_i': result.params,
        't-Values': result.tvalues,
        'p-Values': result.pvalues
       }
data_analysis = pd.DataFrame(data)
data_analysis.round(4) # Round values in table to 4-decimal places
```

Out[71]:

| | Coefficient Beta_i | t-Values | p-Values |
|---|---|---|---|
| **0** | -0.0651 | -0.5319 | 0.5960 |
| **1** | 1.0730 | 5.8018 | 0.0000 |
| **2** | -1.9083 | -19.1667 | 0.0000 |
| **3** | -0.0195 | -0.3136 | 0.7545 |

In [72]:

```
# --------------------- MODEL 4 -----------------
# Make dataset of desired predictors
predictors = {
        'x': x,
        'x^2': x**2,
        'x^3': x**3,
        'x^4': x**4,
       }
data = pd.DataFrame(predictors)
data = np.asarray(data)
data = sm.add_constant(data)


# Create the Multiple Linear Regression Model and fit it
MLRmodel = sm.OLS(y, data)
result = MLRmodel.fit()

# Create table of model prediction results
data = {'Coefficient Beta_i': result.params,
        't-Values': result.tvalues,
        'p-Values': result.pvalues
       }
data_analysis = pd.DataFrame(data)
data_analysis.round(4) # Round values in table to 4-decimal places
```

Out[72]:

| | Coefficient Beta_i | t-Values | p-Values |
|---|---|---|---|
| **0** | -0.0403 | -0.3081 | 0.7587 |
| **1** | 1.1433 | 5.0937 | 0.0000 |
| **2** | -2.0016 | -10.2573 | 0.0000 |
| **3** | -0.0650 | -0.6319 | 0.5290 |
| **4** | 0.0290 | 0.5565 | 0.5792 |

In [ ]: