In [1]:
```
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
```

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

Out[2]:

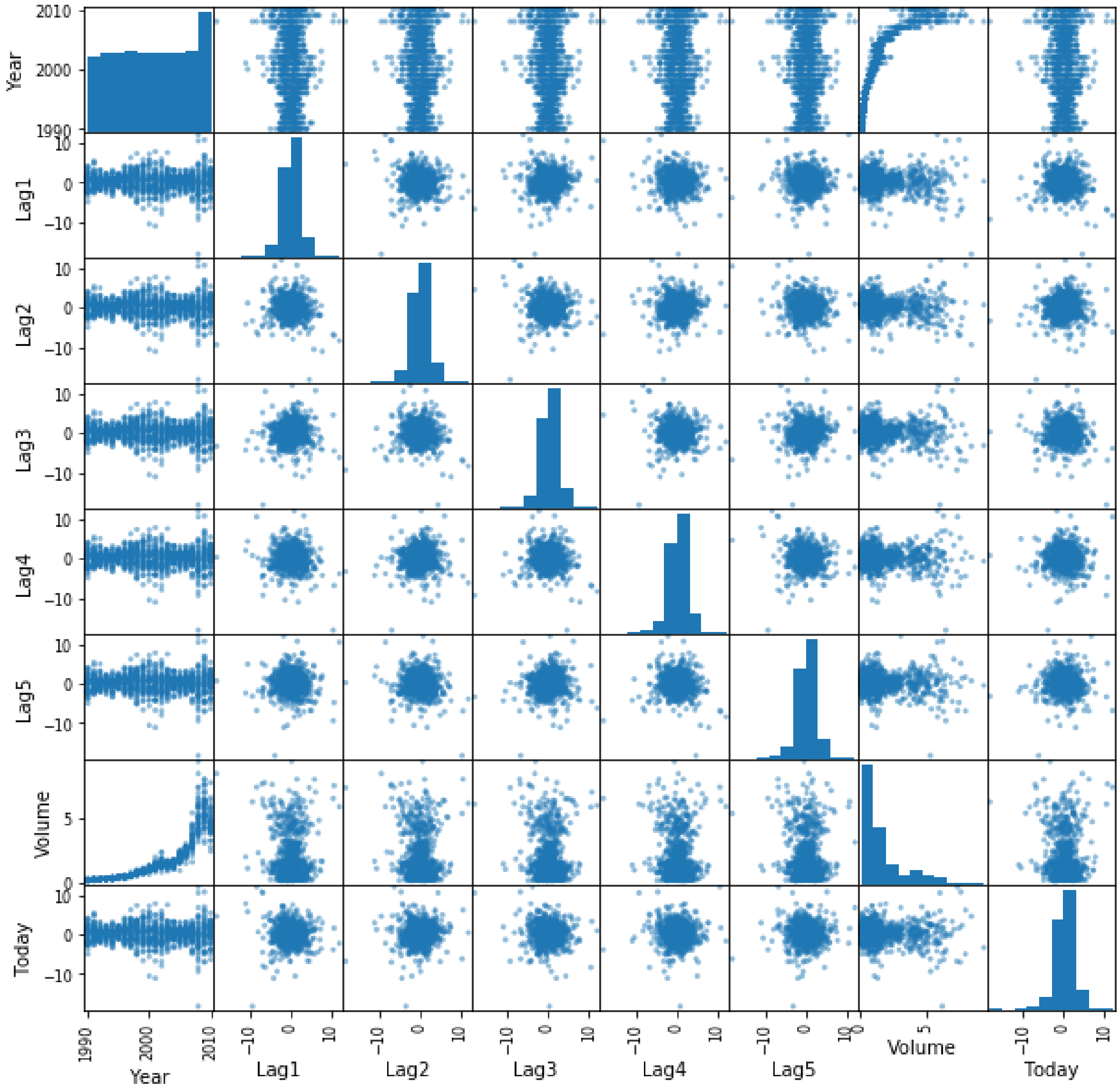
	Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Today	Direction
0	1990	0.816	1.572	-3.936	-0.229	-3.484	0.154976	-0.270	Down
1	1990	-0.270	0.816	1.572	-3.936	-0.229	0.148574	-2.576	Down
2	1990	-2.576	-0.270	0.816	1.572	-3.936	0.159837	3.514	Up
3	1990	3.514	-2.576	-0.270	0.816	1.572	0.161630	0.712	Up
4	1990	0.712	3.514	-2.576	-0.270	0.816	0.153728	1.178	Up

In [3]:
```
# df['Direction'].value_counts()
```

Problem a

In [4]:

```
# Creates Scatterplot Matrix with the quantitative variables
columns = ['Year', 'Lag1', 'Lag2', 'Lag3', 'Lag4', 'Lag5', 'Volume', 'Today']
pd.plotting.scatter_matrix(df[columns], figsize=(10,10)); # Semicolon used to
 suppress array output!
```



Problem b

In [5]:

```python
# Extract only the necessary variables
columns = ['Lag1', 'Lag2', 'Lag3', 'Lag4', 'Lag5', 'Volume', 'Direction']
df_copy = df[columns].copy()

# Replace all 'Up' and 'Down' with 1 and 0 respectively
mapping = {'Up': 1.0, 'Down': 0.0}
df_copy['Direction'] = df_copy['Direction'].replace(mapping)

# Extract "Direction" column as response, and Lags and Volume columns as predictors
y_true = np.asarray(df_copy['Direction'])
X_var = np.asarray(df_copy.iloc[:,:6])
X_design = sm.add_constant(X_var)

# Create array of column names
column_names = list(df_copy.iloc[:,:6].columns)
column_names.insert(0, 'Intercept')

# Create the Logistic Regression Model and fit it
MLRmodel = sm.Logit(y_true, X_design)
fitted_model = MLRmodel.fit()

# Calculate the z-statistics
cov = fitted_model.cov_params() # Model's estimate of the covariance matrix
std_err = np.sqrt(np.diag(cov)) # standard errors = square roots of the variances (diagonal of covariance matrix)
z_values = fitted_model.params / std_err # z-values are coefficients divided by their standard errors

# Create table of model prediction results
data = {'Attributes': column_names,
        'Coefficient Beta_i': fitted_model.params,
        'z-Values': z_values,
        'p-Values': fitted_model.pvalues
       }
ModelResults = pd.DataFrame(data)
ModelResults.round(4) # Round values in table to 4-decimal places
```

```
Optimization terminated successfully.
         Current function value: 0.682441
         Iterations 4
```

Out[5]:

	Attributes	Coefficient Beta_i	z-Values	p-Values
0	Intercept	0.2669	3.1056	0.0019
1	Lag1	-0.0413	-1.5626	0.1181
2	Lag2	0.0584	2.1754	0.0296
3	Lag3	-0.0161	-0.6024	0.5469
4	Lag4	-0.0278	-1.0501	0.2937
5	Lag5	-0.0145	-0.5485	0.5833
6	Volume	-0.0227	-0.6163	0.5377

Problem c

In [6]:

```
y_pred = fitted_model.predict(X_design)
```

In [7]:

```
y_true
```

Out[7]:

```
array([0., 0., 1., ..., 1., 1., 1.])
```

In [8]:

```
y_pred = np.rint(y_pred)
y_pred
```

Out[8]:

```
array([1., 1., 1., ..., 1., 1., 1.])
```

In [9]:

```
num_TP = 0
num_FN = 0
num_FP = 0
num_TN = 0
for actual, pred in zip(y_true, y_pred):
    if (actual - pred) == 0.0:
        if actual == 1:
            num_TP = num_TP + 1
        else :
            num_TN = num_TN + 1
    elif (actual - pred) < 0:
        num_FP = num_FP + 1
    else :
        num_FN = num_FN + 1

print("Number of True Positives: " + repr(num_TP))
print("Number of False Negatives: " + repr(num_FN))
print("Number of False Positives: " + repr(num_FP))
print("Number of True Negatives: " + repr(num_TN))
```

```
Number of True Positives: 557
Number of False Negatives: 48
Number of False Positives: 430
Number of True Negatives: 54
```

Problem d

In [12]:

```
from sklearn.linear_model import LogisticRegression
# Extract only the necessary variables
columns = ['Year', 'Lag2', 'Direction']
df_copy = df[columns].copy()

# Replace all 'Up' and 'Down' with 1 and 0 respectively
mapping = {'Up': 1.0, 'Down': 0.0}
df_copy['Direction'] = df_copy['Direction'].replace(mapping)


# ----------------------- OBTAINING TRAINING AND TESTING DATASETS ---------------------------------
# Select only Datapoints from 1990 to 2008 for training
training_data = df_copy.loc[(df_copy['Year'] >= 1990) & (df_copy['Year'] <= 2008)]

# Select only Datapoints from 2009 to 2010 for testing
test_data = df_copy.loc[(df_copy['Year'] >= 2009) & (df_copy['Year'] <= 2010)]

# TRAINING: Extract "Direction" column as response, and Lag2 column as predictors
y_true_training = np.asarray(training_data['Direction'])
X_var_training = np.reshape(np.asarray(training_data['Lag2']), (-1,1))


# TESTING: Extract "Direction" column as response, and Lag2 column as predictors
y_true_testing = np.asarray(test_data['Direction'])
X_var_testing = np.reshape(np.asarray(test_data['Lag2']), (-1,1))



# ----------------------- FITTING THE MODEL ---------------------------------
# Create the K-Nearest Neighbors Classifier Model and fit it
Logit_classifier = LogisticRegression(random_state=0)
Logit_classifier.fit(X_var_training, y_true_training)


# ----------------------- MAKING PREDICTIONS ---------------------------------
y_pred_test = Logit_classifier.predict(X_var_testing)
num_TN, num_FP, num_FN, num_TP = confusion_matrix(y_true_testing, y_pred_test).ravel()
print("Number of True Positives: " + repr(num_TP))
print("Number of False Negatives: " + repr(num_FN))
print("Number of False Positives: " + repr(num_FP))
print("Number of True Negatives: " + repr(num_TN))
```

```
Number of True Positives: 56
Number of False Negatives: 5
Number of False Positives: 34
Number of True Negatives: 9
```

Problem e

In [13]:

```
# Extract only the necessary variables
columns = ['Year', 'Lag2', 'Direction']
df_copy = df[columns].copy()

# Replace all 'Up' and 'Down' with 1 and 0 respectively
mapping = {'Up': 1.0, 'Down': 0.0}
df_copy['Direction'] = df_copy['Direction'].replace(mapping)


# ----------------------- OBTAINING TRAINING AND TESTING DATASETS ---------------------------------
# Select only Datapoints from 1990 to 2008 for training
training_data = df_copy.loc[(df_copy['Year'] >= 1990) & (df_copy['Year'] <= 2008)]

# Select only Datapoints from 2009 to 2010 for testing
test_data = df_copy.loc[(df_copy['Year'] >= 2009) & (df_copy['Year'] <= 2010)]

# TRAINING: Extract "Direction" column as response, and Lag2 column as predictors
y_true_training = np.asarray(training_data['Direction'])
X_var_training = np.reshape(np.asarray(training_data['Lag2']), (-1,1))


# TESTING: Extract "Direction" column as response, and Lag2 column as predictors
y_true_testing = np.asarray(test_data['Direction'])
X_var_testing = np.reshape(np.asarray(test_data['Lag2']), (-1,1))



# ----------------------- FITTING THE MODEL ---------------------------------
# Create the Linear Discriminant Analysis Classifier Model and fit it
LDA_classifier = LinearDiscriminantAnalysis()
LDA_classifier.fit(X_var_training, y_true_training)


# ----------------------- MAKING PREDICTIONS ---------------------------------
y_pred_test = LDA_classifier.predict(X_var_testing)
num_TN, num_FP, num_FN, num_TP = confusion_matrix(y_true_testing, y_pred_test).ravel()
print("Number of True Positives: " + repr(num_TP))
print("Number of False Negatives: " + repr(num_FN))
print("Number of False Positives: " + repr(num_FP))
print("Number of True Negatives: " + repr(num_TN))
```

```
Number of True Positives: 56
Number of False Negatives: 5
Number of False Positives: 34
Number of True Negatives: 9
```

Problem f

In [41]:

```
# Extract only the necessary variables
columns = ['Year', 'Lag2', 'Direction']
df_copy = df[columns].copy()

# Replace all 'Up' and 'Down' with 1 and 0 respectively
mapping = {'Up': 1.0, 'Down': 0.0}
df_copy['Direction'] = df_copy['Direction'].replace(mapping)


# ----------------------- OBTAINING TRAINING AND TESTING DATASETS ------------
---------------------
# Select only Datapoints from 1990 to 2008 for training
training_data = df_copy.loc[(df_copy['Year'] >= 1990) & (df_copy['Year'] <= 20
08)]

# Select only Datapoints from 2009 to 2010 for testing
test_data = df_copy.loc[(df_copy['Year'] >= 2009) & (df_copy['Year'] <= 2010)]

# TRAINING: Extract "Direction" column as response, and Lag2 column as predict
ors
y_true_training = np.asarray(training_data['Direction'])
X_var_training = np.reshape(np.asarray(training_data['Lag2']), (-1,1))


# TESTING: Extract "Direction" column as response, and Lag2 column as predicto
rs
y_true_testing = np.asarray(test_data['Direction'])
X_var_testing = np.reshape(np.asarray(test_data['Lag2']), (-1,1))



# ----------------------- FITTING THE MODEL ---------------------------------
# Create the Quadratic Discriminant Analysis Classifier Model and fit it
QDA_classifier = QuadraticDiscriminantAnalysis()
QDA_classifier.fit(X_var_training, y_true_training)


# ----------------------- MAKING PREDICTIONS ----------------------------------
y_pred_test = QDA_classifier.predict(X_var_testing)
num_TN, num_FP, num_FN, num_TP = confusion_matrix(y_true_testing, y_pred_test)
.ravel()
print("Number of True Positives: " + repr(num_TP))
print("Number of False Negatives: " + repr(num_FN))
print("Number of False Positives: " + repr(num_FP))
print("Number of True Negatives: " + repr(num_TN))
```

```
Number of True Positives: 61
Number of False Negatives: 0
Number of False Positives: 43
Number of True Negatives: 0
```

Problem g

In [15]:

```
# Extract only the necessary variables
columns = ['Year', 'Lag2', 'Direction']
df_copy = df[columns].copy()

# Replace all 'Up' and 'Down' with 1 and 0 respectively
mapping = {'Up': 1.0, 'Down': 0.0}
df_copy['Direction'] = df_copy['Direction'].replace(mapping)


# ----------------------- OBTAINING TRAINING AND TESTING DATASETS ------------
---------------------
# Select only Datapoints from 1990 to 2008 for training
training_data = df_copy.loc[(df_copy['Year'] >= 1990) & (df_copy['Year'] <= 20
08)]

# Select only Datapoints from 2009 to 2010 for testing
test_data = df_copy.loc[(df_copy['Year'] >= 2009) & (df_copy['Year'] <= 2010)]

# TRAINING: Extract "Direction" column as response, and Lag2 column as predict
ors
y_true_training = np.asarray(training_data['Direction'])
X_var_training = np.reshape(np.asarray(training_data['Lag2']), (-1,1))


# TESTING: Extract "Direction" column as response, and Lag2 column as predicto
rs
y_true_testing = np.asarray(test_data['Direction'])
X_var_testing = np.reshape(np.asarray(test_data['Lag2']), (-1,1))



# ----------------------- FITTING THE MODEL ---------------------------------
# Create the K-Nearest Neighbors Classifier Model and fit it
KNN_classifier = KNeighborsClassifier(n_neighbors=1)
KNN_classifier.fit(X_var_training, y_true_training)


# ----------------------- MAKING PREDICTIONS ---------------------------------
y_pred_test = KNN_classifier.predict(X_var_testing)
num_TN, num_FP, num_FN, num_TP = confusion_matrix(y_true_testing, y_pred_test)
.ravel()
print("Number of True Positives: " + repr(num_TP))
print("Number of False Negatives: " + repr(num_FN))
print("Number of False Positives: " + repr(num_FP))
print("Number of True Negatives: " + repr(num_TN))
```

```
Number of True Positives: 30
Number of False Negatives: 31
Number of False Positives: 22
Number of True Negatives: 21
```

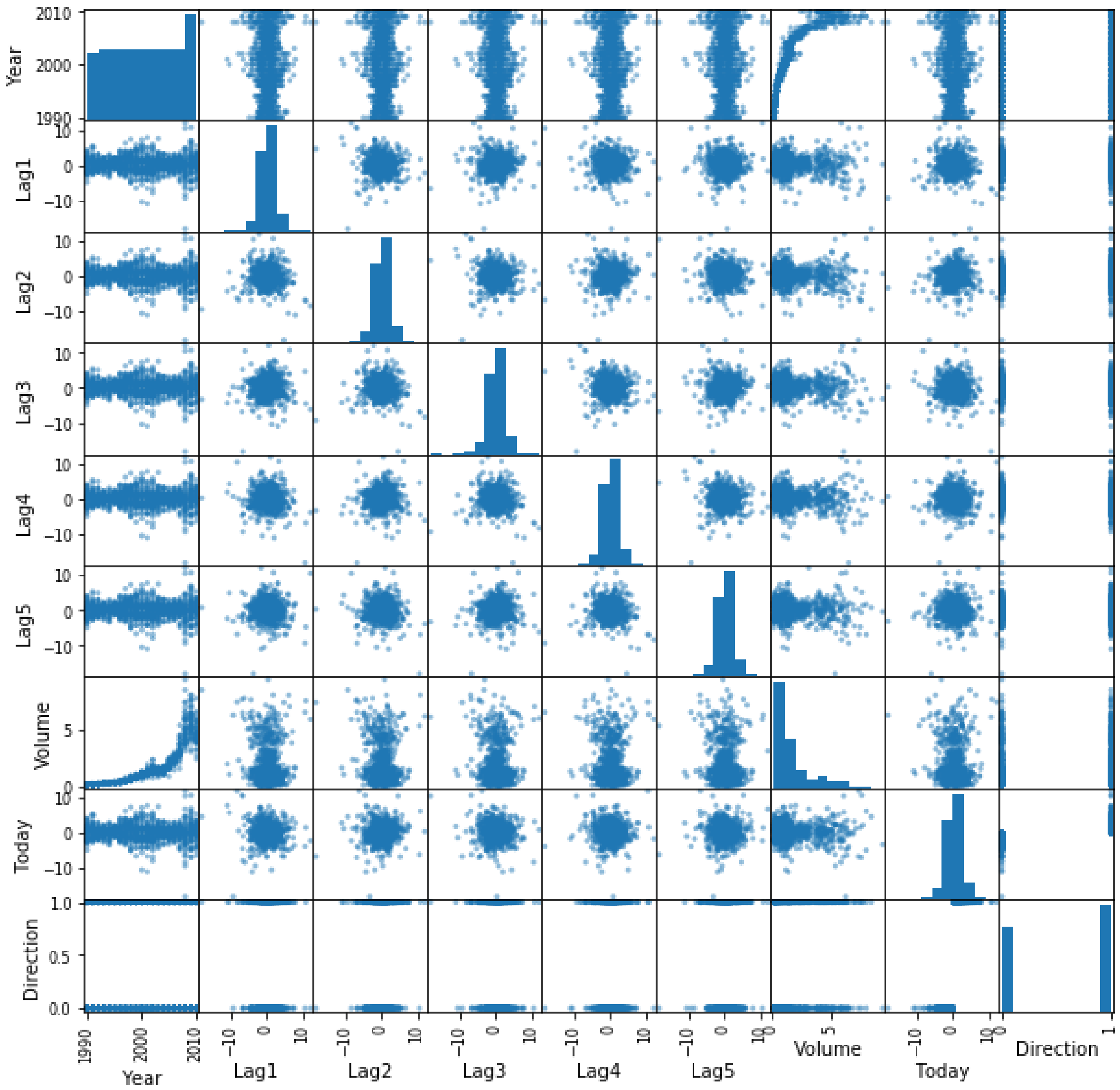
Problem i

In [35]:

```
# Extract only the necessary variables
df_copy = df.copy()

# Replace all 'Up' and 'Down' with 1 and 0 respectively
mapping = {'Up': 1.0, 'Down': 0.0}
df_copy['Direction'] = df_copy['Direction'].replace(mapping)

pd.plotting.scatter_matrix(df_copy, figsize=(10,10));
```



In [50]:

```
# Extract only the necessary variables
df_copy = df.copy()

# Replace all 'Up' and 'Down' with 1 and 0 respectively
mapping = {'Up': 1.0, 'Down': 0.0}
df_copy['Direction'] = df_copy['Direction'].replace(mapping)



# ----------------------- TRANSFORMING DATASET -------------------------------
--
data = {
#         'Lag1': df_copy['Lag1']*df_copy['Lag2']*df_copy['Lag3']*df_copy['Lag
4']*df_copy['Lag5'],
#         'Lag2': df_copy['Lag2'],
#         'Lag3': df_copy['Lag3'],
#         'Lag4': df_copy['Lag4'],
#         'Lag5': df_copy['Lag5'],
        'Today': df_copy['Today'],
        'Year': df_copy['Year'],
        'Volume': np.reciprocal(df_copy['Year']),
        'Direction': df_copy['Direction']
       }
df_copy = pd.DataFrame(data)




# ----------------------- OBTAINING TRAINING AND TESTING DATASETS ------------
---------------------
# Select only Datapoints from 1990 to 2008 for training
training_data = df_copy.loc[(df_copy['Year'] >= 1990) & (df_copy['Year'] <= 20
08)]

# Select only Datapoints from 2009 to 2010 for testing
test_data = df_copy.loc[(df_copy['Year'] >= 2009) & (df_copy['Year'] <= 2010)]



# columns = ['Lag1', 'Lag2', 'Lag3', 'Lag4', 'Lag5', 'Today', 'Volume']
# columns = ['Lag1', 'Today', 'Volume']
columns = ['Today', 'Volume']
# TRAINING: Extract "Direction" column as response, and Lag2 column as predict
ors
y_train = np.asarray(training_data['Direction'])
X_train = np.asarray(training_data[columns])


# TESTING: Extract "Direction" column as response, and Lag2 column as predicto
rs
y_test = np.asarray(test_data['Direction'])
X_test = np.asarray(test_data[columns])



# ----------------------- FITTING THE MODEL ---------------------------------
# Create the K-Nearest Neighbors Classifier Model and fit it
KNN_classifier = KNeighborsClassifier(n_neighbors=5)
KNN_classifier.fit(X_train, y_train)

# # Create the Linear Discriminant Analysis Classifier Model and fit it
# LDA_classifier = LinearDiscriminantAnalysis()
# LDA_classifier.fit(X_train, y_train)

# # Create the Quadratic Discriminant Analysis Classifier Model and fit it
# QDA_classifier = QuadraticDiscriminantAnalysis()
# QDA_classifier.fit(X_train, y_train)


# ----------------------- MAKING PREDICTIONS ---------------------------------
y_pred_test = KNN_classifier.predict(X_test)
num_TN, num_FP, num_FN, num_TP = confusion_matrix(y_test, y_pred_test).ravel()
print("Number of True Positives: " + repr(num_TP))
print("Number of False Negatives: " + repr(num_FN))
print("Number of False Positives: " + repr(num_FP))
print("Number of True Negatives: " + repr(num_TN))
```

```
Number of True Positives: 60
Number of False Negatives: 1
Number of False Positives: 0
Number of True Negatives: 43
```

```
In [ ]:
```