In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

sns.set()

abalone_names = ('Sex',
                 'Length',
                 'Diameter',
                 'Height',
                 'Whole weight',
                 'Shucked weight',
                 'Viscera weight',
                 'Shell weight',
                 'Rings')
```

In [2]:

```
abalone_df = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases
                         names=abalone_names)
abalone_df.head()
```

Out[2]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

In [3]:

```
abalone_df.shape
```

Out[3]:

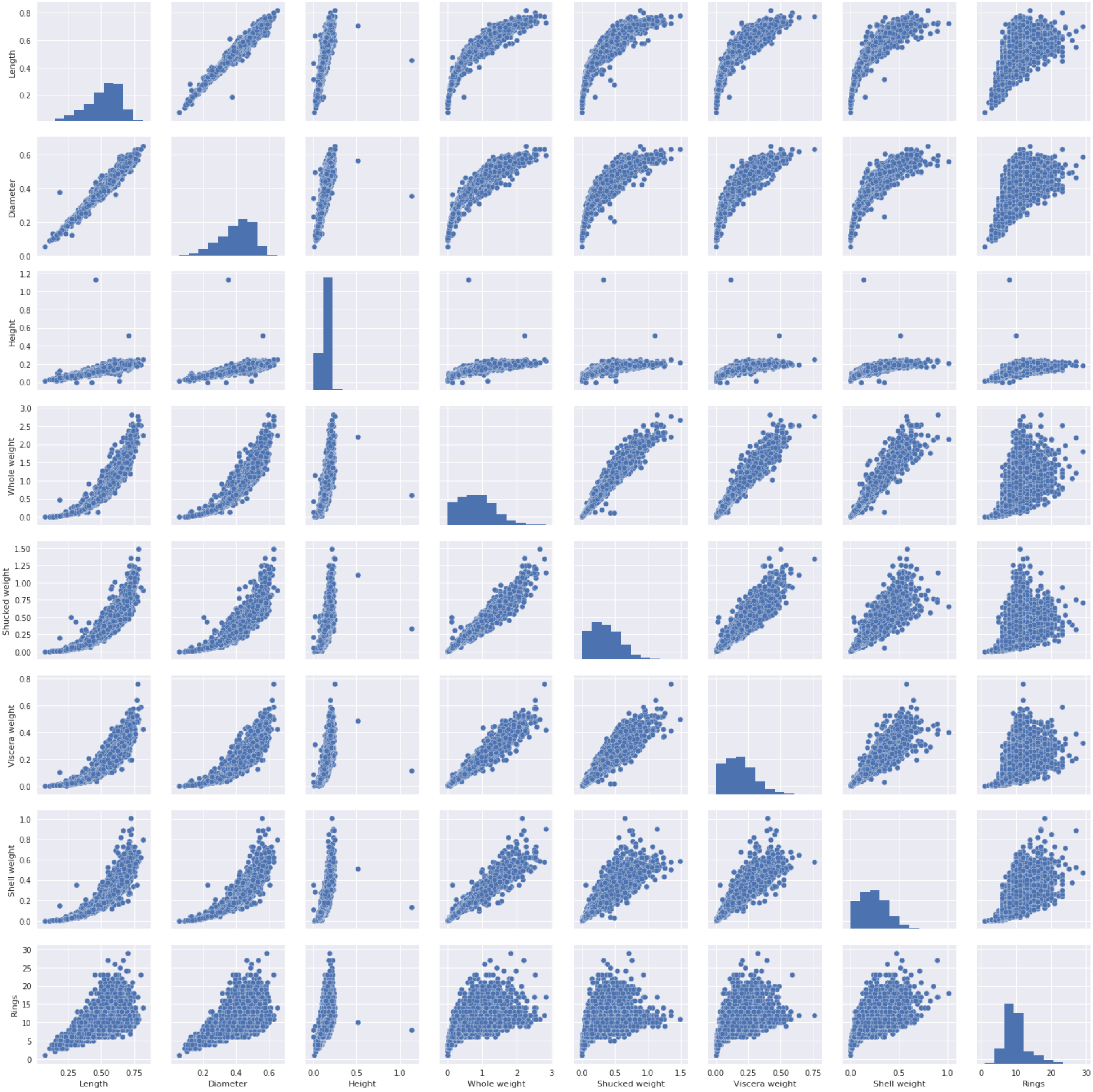
```
(4177, 9)
```

In [4]:

```
abalone_df = abalone_df.drop('Sex', axis=1)
```

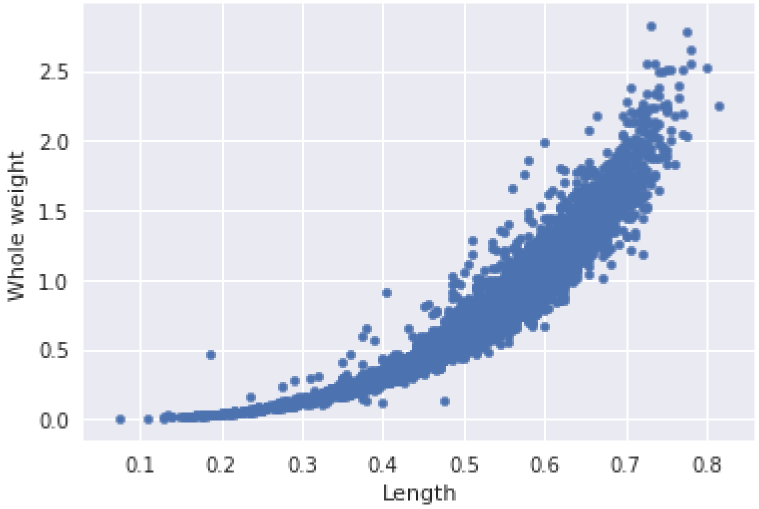
In [5]:

```
sns.pairplot(abalone_df)
plt.show()
```



In [6]:

```
abalone_df.plot.scatter('Length', 'Whole weight')
plt.show()
```



In [7]:

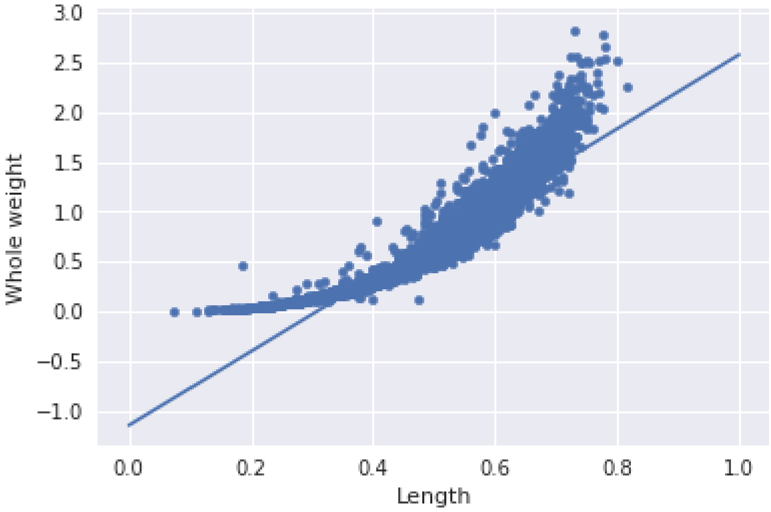
```
from sklearn.svm import SVR

reg = SVR(kernel='linear')
reg.fit(abalone_df[['Length']], abalone_df['Whole weight'])
```

Out[7]:

```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='auto',
  kernel='linear', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

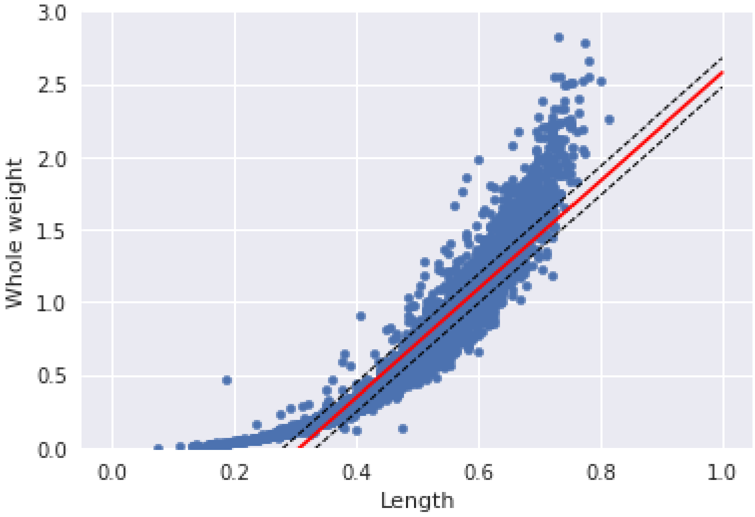
In [8]:

```
x_reg = np.linspace(0, 1, 100)
x_reg = np.expand_dims(x_reg, axis=1)
y_reg = reg.predict(x_reg)

abalone_df.plot.scatter('Length', 'Whole weight')
plt.plot(x_reg, y_reg)
plt.show()
```



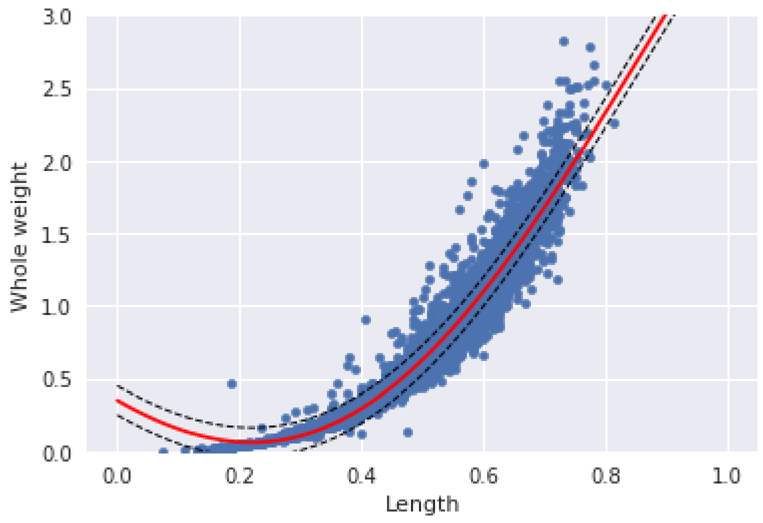
In [9]:

```
abalone_df.plot.scatter('Length', 'Whole weight')
plt.plot(x_reg, y_reg, c='r')
plt.plot(x_reg, y_reg + reg.epsilon, '--', c='k', linewidth=1)
plt.plot(x_reg, y_reg - reg.epsilon, '--', c='k', linewidth=1)

plt.ylim(0, 3)
plt.show()
```



In [10]:

```
reg = SVR(kernel='rbf')
reg.fit(abalone_df[['Length']], abalone_df['Whole weight'])
y_reg = reg.predict(x_reg)

abalone_df.plot.scatter('Length', 'Whole weight')
plt.plot(x_reg, y_reg, c='r')

plt.plot(x_reg, y_reg + reg.epsilon, '--', c='k', linewidth=1)
plt.plot(x_reg, y_reg - reg.epsilon, '--', c='k', linewidth=1)

plt.ylim(0, 3)
plt.show()
```

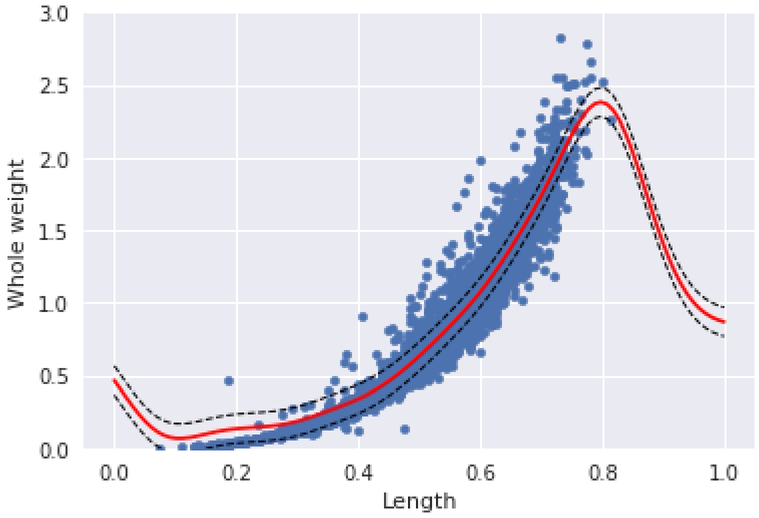


In [11]:

```
def svr_rbf_experiment(gamma):
    reg = SVR(kernel='rbf', gamma=gamma)
    reg.fit(abalone_df[['Length']], abalone_df['Whole weight'])
    y_reg = reg.predict(x_reg)

    abalone_df.plot.scatter('Length', 'Whole weight')
    plt.plot(x_reg, y_reg, c='r')

    plt.plot(x_reg, y_reg + reg.epsilon, '--', c='k', linewidth=1)
    plt.plot(x_reg, y_reg - reg.epsilon, '--', c='k', linewidth=1)
    plt.ylim(0, 3)
```

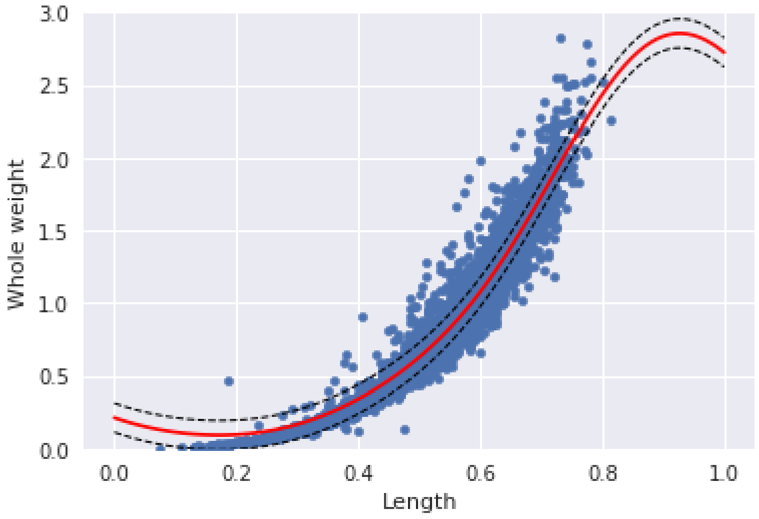
In [12]:

```
svr_rbf_experiment(100)
plt.show()
```



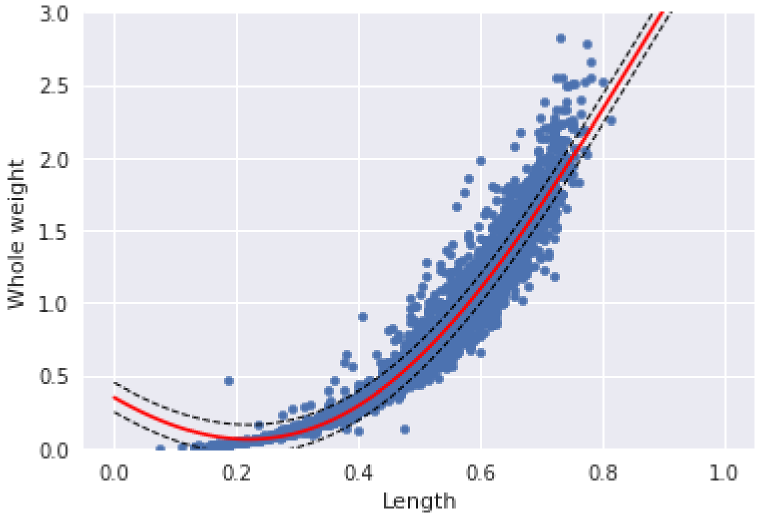
In [13]:

```
svr_rbf_experiment(10)
plt.show()
```



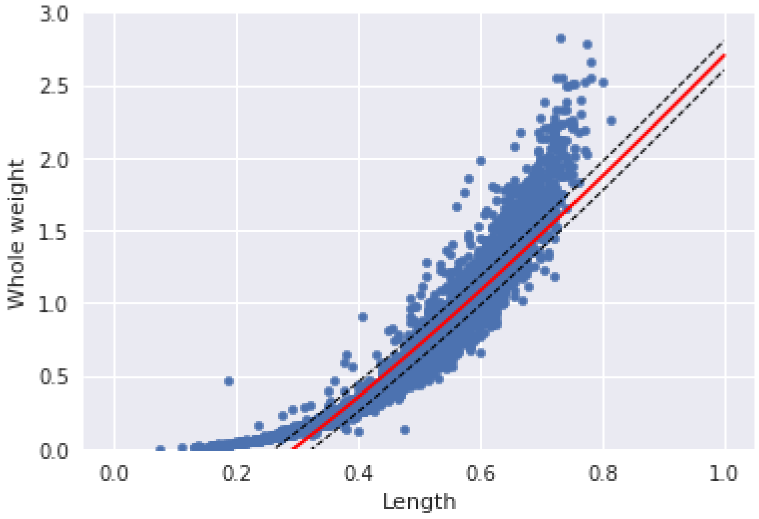
In [14]:

```
svr_rbf_experiment(1)
plt.show()
```



In [15]:

```
svr_rbf_experiment(.1)
plt.show()
```

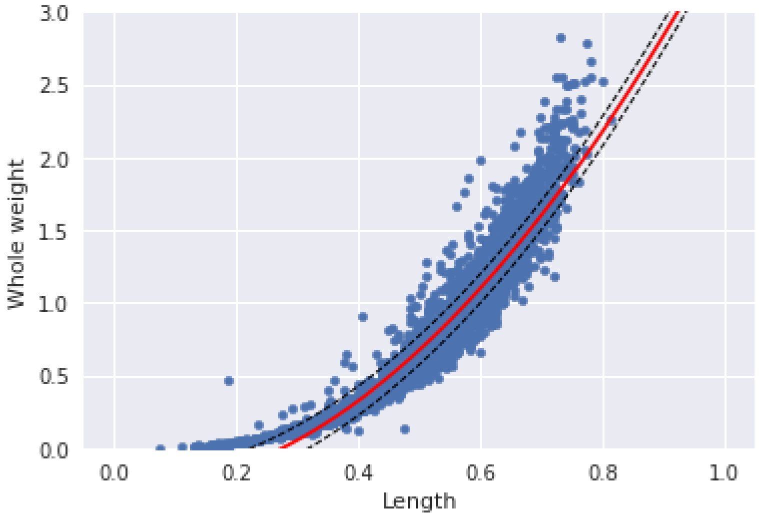


In [16]:

```
reg = SVR(kernel='poly', degree=2)
reg.fit(abalone_df[['Length']], abalone_df['Whole weight'])
y_reg = reg.predict(x_reg)

abalone_df.plot.scatter('Length', 'Whole weight')
plt.plot(x_reg, y_reg, c='r')

plt.plot(x_reg, y_reg + reg.epsilon, '--', c='k', linewidth=1)
plt.plot(x_reg, y_reg - reg.epsilon, '--', c='k', linewidth=1)
plt.ylim(0, 3)
```

Out[16]:

```
(0, 3)
```

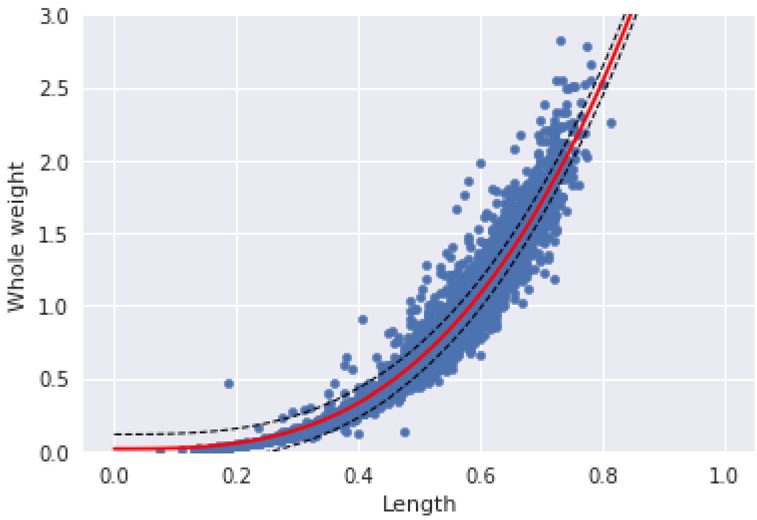


In [17]:

```
reg = SVR(kernel='poly', degree=3)
reg.fit(abalone_df[['Length']], abalone_df['Whole weight'])
y_reg = reg.predict(x_reg)

abalone_df.plot.scatter('Length', 'Whole weight')
plt.plot(x_reg, y_reg, c='r')

plt.plot(x_reg, y_reg + reg.epsilon, '--', c='k', linewidth=1)
plt.plot(x_reg, y_reg - reg.epsilon, '--', c='k', linewidth=1)
plt.ylim(0, 3)
```

Out[17]:

```
(0, 3)
```

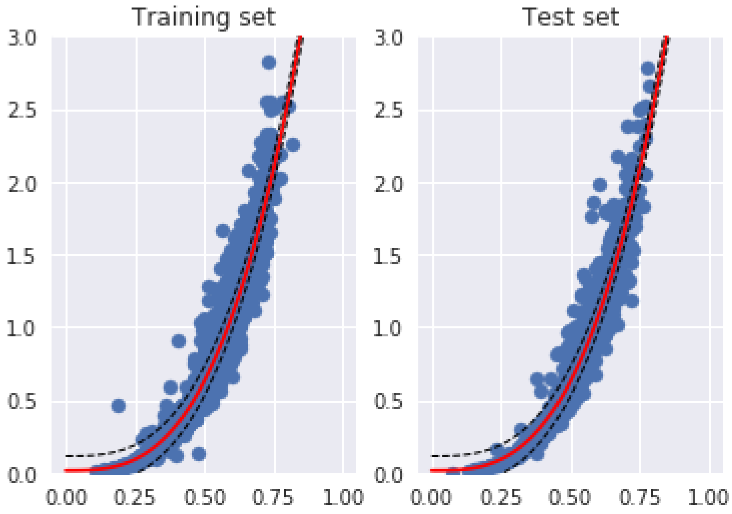


In [18]:

```
from sklearn.model_selection import train_test_split

x_train, x_test, \
y_train, y_test = train_test_split(abalone_df[['Length']],
                                   abalone_df['Whole weight'],
                                   test_size=0.3)
```

In [19]:

```
reg.fit(x_train, y_train)
y_pred = reg.predict(x_test)
y_reg = reg.predict(x_reg)

plt.subplot(1, 2, 1)
plt.scatter(x_train, y_train)
plt.plot(x_reg, y_reg, c='r')

plt.plot(x_reg, y_reg + reg.epsilon, '--', c='k', linewidth=1)
plt.plot(x_reg, y_reg - reg.epsilon, '--', c='k', linewidth=1)

plt.ylim(0, 3)
plt.title('Training set')

plt.subplot(1, 2, 2)
plt.scatter(x_test, y_test)
plt.plot(x_reg, y_reg, c='r')

plt.plot(x_reg, y_reg + reg.epsilon, '--', c='k', linewidth=1)
plt.plot(x_reg, y_reg - reg.epsilon, '--', c='k', linewidth=1)

plt.ylim(0, 3)
plt.title('Test set')

plt.show()
```



In [20]:

```
from sklearn.metrics import r2_score

print(r2_score(y_test, y_pred))
```

```
0.9233887547300553
```

In [21]:

```
c_vals = [1, 10, 100]
eps_vals = [.1, 1]

tuned_parameters = [{'kernel': ['rbf'],    'gamma': [1, 5],
                                           'C': c_vals,
                                           'epsilon': eps_vals},
                    {'kernel': ['linear'], 'C': c_vals,
                                           'epsilon': eps_vals},
                    {'kernel': ['poly'],   'degree': [3, 4],
                                           'C': c_vals,
                                           'epsilon': eps_vals}]
```

In [22]:

```
from sklearn.model_selection import GridSearchCV

reg = GridSearchCV(SVR(), tuned_parameters, cv=3, verbose=1)
reg.fit(x_train, y_train)
```

```
Fitting 3 folds for each of 30 candidates, totalling 90 fits

[Parallel(n_jobs=1)]: Done  90 out of  90 | elapsed:    4.7s finished
```

Out[22]:

```
GridSearchCV(cv=3, error_score='raise',
       estimator=SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamm
a='auto',
  kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False),
       fit_params=None, iid=True, n_jobs=1,
       param_grid=[{'kernel': ['rbf'], 'gamma': [1, 5], 'C': [1, 10, 100], 'epsilo
n': [0.1, 1]}, {'kernel': ['linear'], 'C': [1, 10, 100], 'epsilon': [0.1, 1]}, {'k
ernel': ['poly'], 'degree': [3, 4], 'C': [1, 10, 100], 'epsilon': [0.1, 1]}],
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=1)
```

In [23]:

```
print(reg.best_params_)
```

```
{'C': 100, 'degree': 3, 'epsilon': 0.1, 'kernel': 'poly'}
```

In [24]:

```
means = reg.cv_results_['mean_test_score']
for mean, params in zip(means, reg.cv_results_['params']):
    print('Average score for {}: {:.2f}'.format(params, mean))
```

```
Average score for {'C': 1, 'epsilon': 0.1, 'gamma': 1, 'kernel': 'rbf'}: 0.93
Average score for {'C': 1, 'epsilon': 0.1, 'gamma': 5, 'kernel': 'rbf'}: 0.93
Average score for {'C': 1, 'epsilon': 1, 'gamma': 1, 'kernel': 'rbf'}: -0.52
Average score for {'C': 1, 'epsilon': 1, 'gamma': 5, 'kernel': 'rbf'}: -0.45
Average score for {'C': 10, 'epsilon': 0.1, 'gamma': 1, 'kernel': 'rbf'}: 0.93
Average score for {'C': 10, 'epsilon': 0.1, 'gamma': 5, 'kernel': 'rbf'}: 0.93
Average score for {'C': 10, 'epsilon': 1, 'gamma': 1, 'kernel': 'rbf'}: -0.34
Average score for {'C': 10, 'epsilon': 1, 'gamma': 5, 'kernel': 'rbf'}: -0.45
Average score for {'C': 100, 'epsilon': 0.1, 'gamma': 1, 'kernel': 'rbf'}: 0.93
Average score for {'C': 100, 'epsilon': 0.1, 'gamma': 5, 'kernel': 'rbf'}: 0.93
Average score for {'C': 100, 'epsilon': 1, 'gamma': 1, 'kernel': 'rbf'}: -0.34
Average score for {'C': 100, 'epsilon': 1, 'gamma': 5, 'kernel': 'rbf'}: -0.45
Average score for {'C': 1, 'epsilon': 0.1, 'kernel': 'linear'}: 0.86
Average score for {'C': 1, 'epsilon': 1, 'kernel': 'linear'}: -0.56
Average score for {'C': 10, 'epsilon': 0.1, 'kernel': 'linear'}: 0.86
Average score for {'C': 10, 'epsilon': 1, 'kernel': 'linear'}: -0.34
Average score for {'C': 100, 'epsilon': 0.1, 'kernel': 'linear'}: 0.86
Average score for {'C': 100, 'epsilon': 1, 'kernel': 'linear'}: -0.31
Average score for {'C': 1, 'degree': 3, 'epsilon': 0.1, 'kernel': 'poly'}: 0.93
Average score for {'C': 1, 'degree': 3, 'epsilon': 1, 'kernel': 'poly'}: -0.22
Average score for {'C': 1, 'degree': 4, 'epsilon': 0.1, 'kernel': 'poly'}: 0.93
Average score for {'C': 1, 'degree': 4, 'epsilon': 1, 'kernel': 'poly'}: -0.10
Average score for {'C': 10, 'degree': 3, 'epsilon': 0.1, 'kernel': 'poly'}: 0.94
Average score for {'C': 10, 'degree': 3, 'epsilon': 1, 'kernel': 'poly'}: -0.29
Average score for {'C': 10, 'degree': 4, 'epsilon': 0.1, 'kernel': 'poly'}: 0.93
Average score for {'C': 10, 'degree': 4, 'epsilon': 1, 'kernel': 'poly'}: -0.20
Average score for {'C': 100, 'degree': 3, 'epsilon': 0.1, 'kernel': 'poly'}: 0.94
Average score for {'C': 100, 'degree': 3, 'epsilon': 1, 'kernel': 'poly'}: -0.29
Average score for {'C': 100, 'degree': 4, 'epsilon': 0.1, 'kernel': 'poly'}: 0.93
Average score for {'C': 100, 'degree': 4, 'epsilon': 1, 'kernel': 'poly'}: -0.20
```

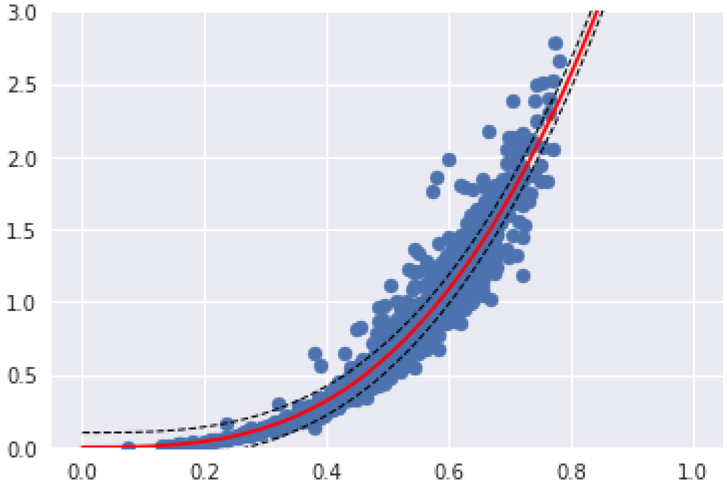
In [25]:

```
from sklearn.metrics import r2_score

y_pred = reg.predict(x_test)
print(r2_score(y_test, y_pred))
```

```
0.9237840432121829
```

In [26]:

```
plt.scatter(x_test, y_test)

y_reg = reg.predict(x_reg)
plt.plot(x_reg, y_reg, c='r')

epsilon = reg.best_params_['epsilon']

plt.plot(x_reg, y_reg + epsilon, '--', c='k', linewidth=1)
plt.plot(x_reg, y_reg - epsilon, '--', c='k', linewidth=1)

plt.ylim(0, 3)
plt.show()
```



In [27]:

```
tuned_parameters = [{'kernel': ['rbf'],    'gamma': [1, 5, 10],
                                            'C': c_vals,
                                            'epsilon': eps_vals},
                    {'kernel': ['linear'], 'C': c_vals,
                                            'epsilon': eps_vals}]


x_train, x_test, \
y_train, y_test = train_test_split(abalone_df.drop(['Whole weight'], axis=1),
                                   abalone_df['Whole weight'],
                                   test_size=0.3)
```

In [28]:

```
reg = GridSearchCV(SVR(), tuned_parameters, cv=3, verbose=1)
reg.fit(x_train, y_train)
```

```
Fitting 3 folds for each of 24 candidates, totalling 72 fits

[Parallel(n_jobs=1)]: Done  72 out of  72 | elapsed:    9.5s finished
```

Out[28]:

```
GridSearchCV(cv=3, error_score='raise',
       estimator=SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamm
a='auto',
  kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False),
       fit_params=None, iid=True, n_jobs=1,
       param_grid=[{'kernel': ['rbf'], 'gamma': [1, 5, 10], 'C': [1, 10, 100], 'ep
silon': [0.1, 1]}, {'kernel': ['linear'], 'C': [1, 10, 100], 'epsilon': [0.1,
1]}],
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=1)
```

In [29]:

```
y_pred = reg.predict(x_test)
print(r2_score(y_test, y_pred))
```

```
0.9897509549884517
```

In [30]:

```
from sklearn.kernel_ridge import KernelRidge

alpha_vals = [.1, 1, 10]

tuned_parameters = [{'kernel': ['rbf'],     'gamma': [.1, 1, 10, 100],
                                             'alpha': alpha_vals},
                    {'kernel': ['linear'], 'alpha': alpha_vals}]

reg = GridSearchCV(KernelRidge(), tuned_parameters, cv=3, verbose=1)
reg.fit(x_train, y_train)
```

```
Fitting 3 folds for each of 15 candidates, totalling 45 fits

[Parallel(n_jobs=1)]: Done  45 out of  45 | elapsed:   16.8s finished
```

Out[30]:

```
GridSearchCV(cv=3, error_score='raise',
       estimator=KernelRidge(alpha=1, coef0=1, degree=3, gamma=None, kernel='linea
r',
      kernel_params=None),
       fit_params=None, iid=True, n_jobs=1,
       param_grid=[{'kernel': ['rbf'], 'gamma': [0.1, 1, 10, 100], 'alpha': [0.1,
1, 10]}, {'kernel': ['linear'], 'alpha': [0.1, 1, 10]}],
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=1)
```

In [31]:

```
y_pred = reg.predict(x_test)
print(r2_score(y_test, y_pred))
```

```
0.99256974758035
```

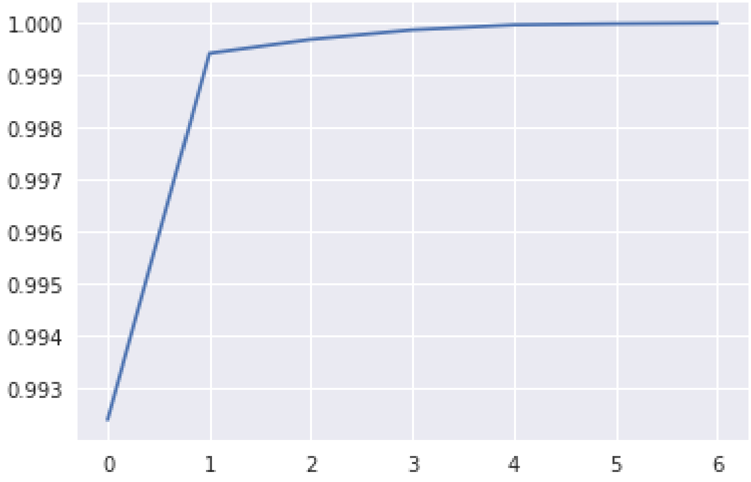
In [32]:

```
import time
from sklearn.decomposition import PCA

num_components = 7

time_start = time.time()
pca = PCA(n_components=num_components)
pca.fit(x_train)
print('PCA with 8 components done!'
      ' Time elapsed: {:.2f} seconds'.format(time.time()-time_start))
```

```
PCA with 8 components done! Time elapsed: 0.00 seconds
```

In [33]:

```
plt.plot(range(num_components), pca.explained_variance_ratio_.cumsum())
plt.show()
```



In [86]:

```
num_components = 3

pca = PCA(n_components=num_components)
pca.fit(x_train)

x_train_3d = pca.transform(x_train)
x_test_3d = pca.transform(x_test)
```

In [87]:

```
tuned_parameters = [{'kernel': ['rbf'],     'gamma': [1, 5, 10],
                                            'C': c_vals,
                                            'epsilon': eps_vals}]

reg = GridSearchCV(SVR(), tuned_parameters, cv=3, verbose=1)
reg.fit(x_train_3d, y_train)
```

```
Fitting 3 folds for each of 18 candidates, totalling 54 fits

[Parallel(n_jobs=1)]: Done  54 out of  54 | elapsed:    3.1s finished
```

Out[87]:

```
GridSearchCV(cv=3, error_score='raise',
       estimator=SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamm
a='auto',
  kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False),
       fit_params=None, iid=True, n_jobs=1,
       param_grid=[{'kernel': ['rbf'], 'gamma': [1, 5, 10], 'C': [1, 10, 100], 'ep
silon': [0.1, 1]}],
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=1)
```

In [88]:

```
reg.best_params_
```

Out[88]:

```
{'C': 10, 'epsilon': 0.1, 'gamma': 1, 'kernel': 'rbf'}
```

In [90]:

```
y_pred = reg.predict(x_test_3d)
print(r2_score(y_test, y_pred))
```

```
0.9760486431716043
```

In [91]:

```
tuned_parameters = [{'kernel': ['rbf'],    'gamma': [.1, 1, 10, 100],
                                           'alpha': alpha_vals}]

reg = GridSearchCV(KernelRidge(), tuned_parameters, cv=3, verbose=1)
reg.fit(x_train_3d, y_train)
```

```
Fitting 3 folds for each of 12 candidates, totalling 36 fits

[Parallel(n_jobs=1)]: Done  36 out of  36 | elapsed:   18.8s finished
```

Out[91]:

```
GridSearchCV(cv=3, error_score='raise',
       estimator=KernelRidge(alpha=1, coef0=1, degree=3, gamma=None, kernel='linea
r',
      kernel_params=None),
       fit_params=None, iid=True, n_jobs=1,
       param_grid=[{'kernel': ['rbf'], 'gamma': [0.1, 1, 10, 100], 'alpha': [0.1,
1, 10]}],
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=1)
```

In [92]:

```
y_pred = reg.predict(x_test_3d)
print(r2_score(y_test, y_pred))
```

```
0.9838151786529818
```

In [93]:

```
reg.best_params_
```

Out[93]:

```
{'alpha': 0.1, 'gamma': 0.1, 'kernel': 'rbf'}
```

In [ ]:

```
```