QUIZ PEMBELAJARAN MESIN
NAIVE BAYES DAN SUPPORT VECTOR MACHINE

VINCENT MICHAEL SUTANTO
16/398531/PA/17492

PROGRAM STUDI ILMU KOMPUTER
DEPARTEMEN ILMU KOMPUTER DAN ELEKTRONIKA
UNIVERSITAS GADJAH MADA
YOGYAKARTA
2019

1. NAIVE BAYES

A. MODEL NAIVE BAYES DARI DATA VISIT-NOMINAL.CSV

Bahasa Pemrograman : Python (Jupyter, Scikit-Learn, Pandas, Seaborn)
Repository : https://github.com/vincentmichael089/ML-NB-SVM/blob/master/NaiveBayes.ipynb
Source Code : terlampir (di akhir tugas)
Dataset : Visit-Nominal.csv
Jenis Naive-Bayes : Bernoulli (karena data pada fitur berupa boolean)
Score : 0.56
Cross-Validation Score : 0.52
Model Naive Bayes :

Probability of Class 0

feature 1 : 0.732575714272584
feature 2 : 0.7826070218144325
feature 3 : 0.5337704624000148
feature 4 : 0.6275952022872798
feature 5 : 0.7155617759223357
feature 6 : 0.6093515137984894

Probability of Class 1

feature 1 : 0.6711168360853097
feature 2 : 0.7954032230077729
feature 3 : 0.4465362457883447
feature 4 : 0.7712692781806878
feature 5 : 0.6711168360853097
feature 6 : 0.591444335183008

B. PERHITUNGAN NILAI ATRIBUT MANUAL 2 DATA

Data 1 = [1,0,1,0,1,0]
Data 2 = [1,1,0,1,0,1]

	Home	Browsed	Searched	Prod_A	Prod_B	Prod_C	Visit_Again
count	39	39	39	39	39	39	39
unique	2	2	2	2	2	2	1
top	yes	yes	no	yes	no	no	yes
freq	23	27	24	25	20	22	39

	Home	Browsed	Searched	Prod_A	Prod_B	Prod_C	Visit_Again
count	61	61	61	61	61	61	61
unique	2	2	2	2	2	2	1
top	yes	yes	no	no	yes	no	no
freq	37	45	33	33	36	33	61

P(Visit_Again = Yes) = 39/100
P(Visit_Again = No) = 61/100

P(Home = Yes) = 60/100
P(Home = No) = 40/100
P(Browsed = Yes) = 72/100
P(Browsed = No) = 28/100
P(Searched = Yes) = 43/100
P(Searched = No) = 57/100
P(Prod_A = Yes) = 53/100
P(Prod_A = No) = 47/100
P(Prod_B = Yes) = 55/100
P(Prod_B = No) = 45/100
P(Prod_C = Yes) = 45/100
P(Prod_C = No) = 55/100

P(Home = Yes | Visit_Again = Yes) = 23/39
P(Home = No | Visit_Again = Yes) = 16/39
P(Browsed = Yes | Visit_Again = Yes) = 27/39
P(Browsed = No | Visit_Again = Yes) = 12/39
P(Searched = Yes | Visit_Again = Yes) = 15/39
P(Searched = No | Visit_Again = Yes) = 24/39
P(Prod_A = Yes | Visit_Again = Yes) = 25/39
P(Prod_A = No | Visit_Again = Yes) = 14/39
P(Prod_B = Yes | Visit_Again = Yes) = 19/39
P(Prod_B = No | Visit_Again = Yes) = 20/39
P(Prod_C = Yes | Visit_Again = Yes) = 17/39
P(Prod_C = No | Visit_Again = Yes) = 22/39

P(Home = Yes | Visit_Again = No) = 37/61
P(Home = No | Visit_Again = No) = 24/61
P(Browsed = Yes | Visit_Again = No) = 45/61
P(Browsed = No | Visit_Again = No) = 16/61
P(Searched = Yes | Visit_Again = No) = 28/61
P(Searched = No | Visit_Again = No) = 33/61
P(Prod_A = Yes | Visit_Again = No) = 28/61
P(Prod_A = No | Visit_Again = No) = 33/61
P(Prod_B = Yes | Visit_Again = No) = 36/61
P(Prod_B = No | Visit_Again = No) = 25/61
P(Prod_C = Yes | Visit_Again = No) = 28/61
P(Prod_C = No | Visit_Again = No) = 33/61

Data 1:
P(Visit_Again = Yes | X)
= ((23/39) * (12/39) * (15/39) * (14/39) * (19/39) * (22/39))
/ (0.6 * 0.28 * 0.43 * 0.46 * 0.55 * 0.55)
= 0.6849455815

P(Visit_Again = No | X) =
= ((37/61) * (16/61) * (28/61) *(33/61) * (36/61) * (33/61))
/ (0.6 * 0.28 * 0.43 * 0.46 * 0.55 * 0.55)
= 1.25478743319
Dari data 1 diprediksi Label Visit_Again = No

Data 2:
P(Visit_Again = Yes | X) =
= ((23/39) * (27/39) * (24/39) * (25/39) * (20/39) * (17/39))
/ (0.6 * 0.72 * 0.56 * 0.53 * 0.45 * 0.45)
= 1.38663275371
P(Visit_Again = No | X) =
= ((37/61) * (45/61) * (33/61) * (28/61) * (25/61) * (28/61))
/ (0.6 * 0.72 * 0.56 * 0.53 * 0.45 * 0.45)
= 0.80506800644
Dari data 1 diprediksi Label Visit_Again = Yes

2. SUPPORT VECTOR MACHINE

A. MODEL 5-FOLD CROSS VALIDATION 3 KERNEL DENGAN NILAI C = 0.01, 0.1, 1, 10

Bahasa Pemrograman : Python (Jupyter, Scikit-Learn, Pandas, Seaborn)
Repository : https://github.com/vincentmichael089/ML-NB-SVM/blob/master/SVM.ipynb
Source Code : terlampir (di akhir tugas)
Dataset : Visit-Nominal.csv
Model SVM:

a. RBF

CROSS VALIDATION SCORE RBF C=0.01 : 0.610125313283208
CROSS VALIDATION SCORE RBF C=0.1 : 0.610125313283208
CROSS VALIDATION SCORE RBF C=1 : 0.5095989974937344
CROSS VALIDATION SCORE RBF C=10 : 0.610125313283208

b. SIGMOID

CROSS VALIDATION SCORE SIGMOID C=0.01 : 0.610125313283208
CROSS VALIDATION SCORE SIGMOID C=0.1 : 0.610125313283208
CROSS VALIDATION SCORE SIGMOID C=1 : 0.640125313283208
CROSS VALIDATION SCORE SIGMOID C=10 : 0.46849624060150374

c. LINEAR

CROSS VALIDATION SCORE LINEAR C=0.01 : 0.610125313283208
CROSS VALIDATION SCORE LINEAR C=0.1 : 0.610125313283208
CROSS VALIDATION SCORE LINEAR C=1 : 0.610125313283208
CROSS VALIDATION SCORE LINEAR C=10 : 0.46849624060150374

B. PLOT HASIL AKURASI DAN KESIMPULAN

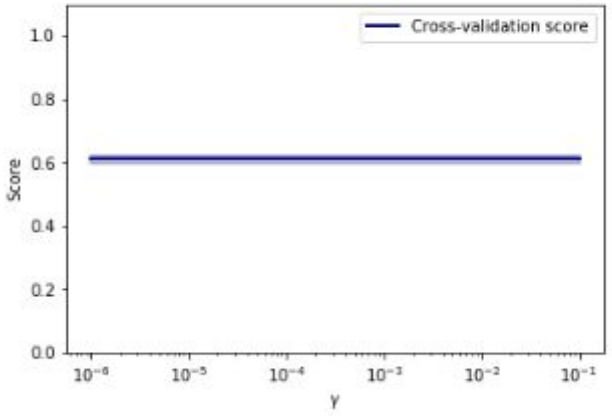
Hasil Cross-Validation dengan akurasi tertinggi :

Model : SIGMOID
C : 1
Akurasi: 0.640125313283208

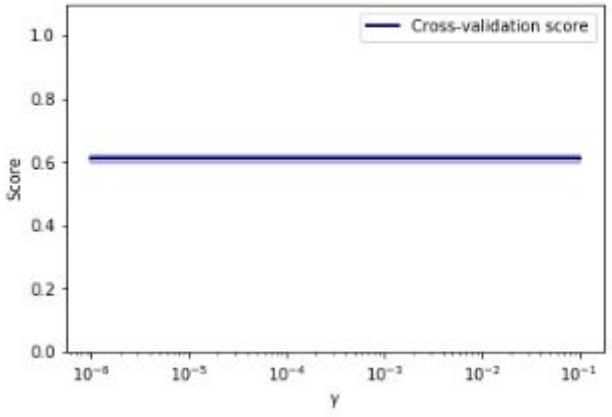
Plot:

a. RBF

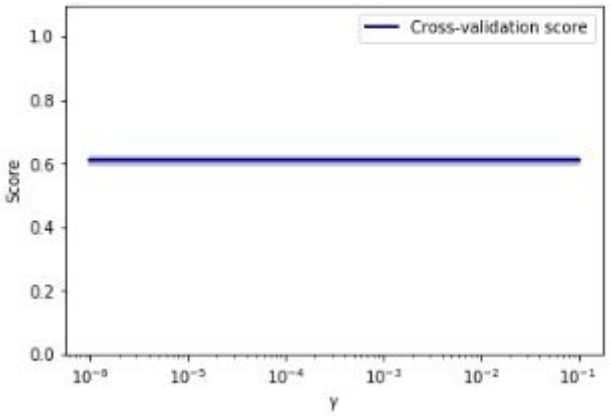
CROSS VALIDATION SCORE RBF C=0.01 : 0.610125313283208



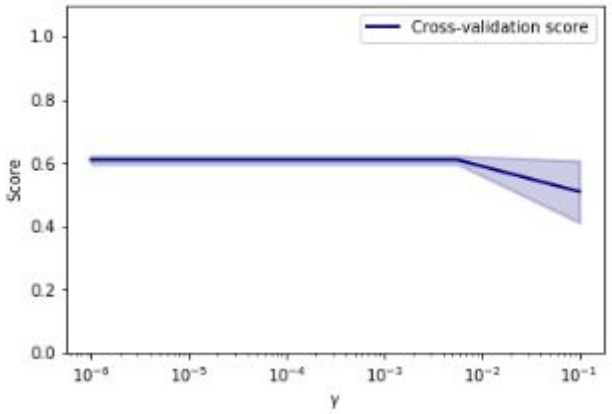
CROSS VALIDATION SCORE RBF C=0.1 : 0.610125313283208



```
CROSS VALIDATION SCORE RBF C=1 :  0.5095989974937344
```

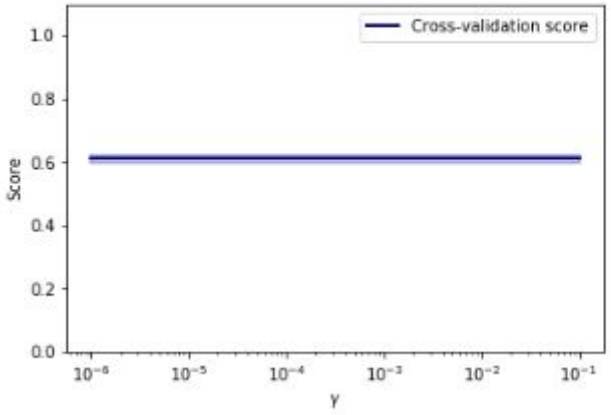



```
CROSS VALIDATION SCORE RBF C=10 :  0.610125313283208
```

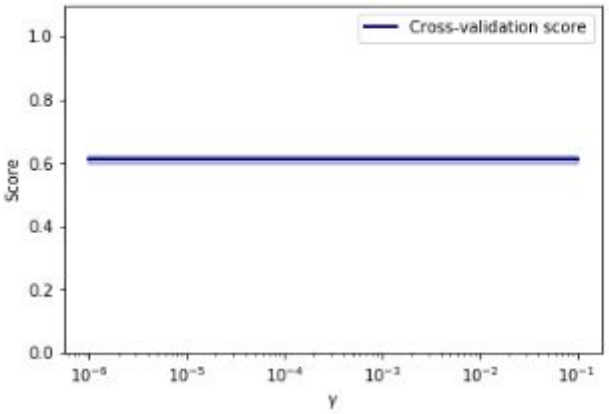


b. SIGMOID

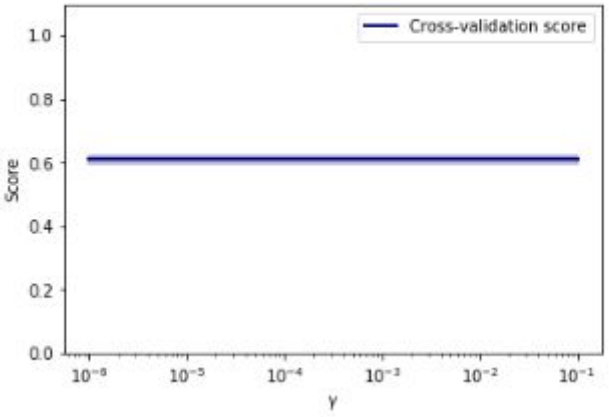
```
CROSS VALIDATION SCORE SIGMOID C=0.01 :  0.610125313283208
```



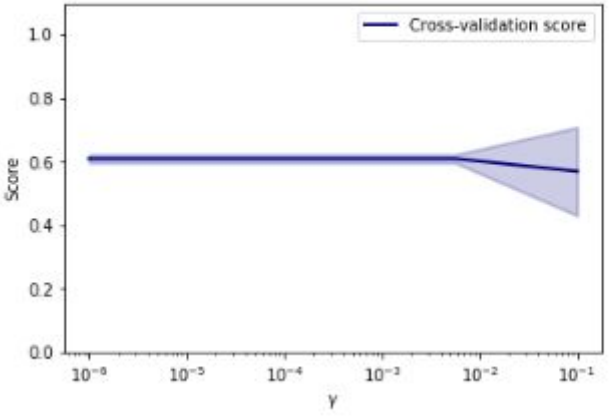
CROSS VALIDATION SCORE SIGMOID C=0.1 : 0.610125313283208



CROSS VALIDATION SCORE SIGMOID C=1 : 0.640125313283208

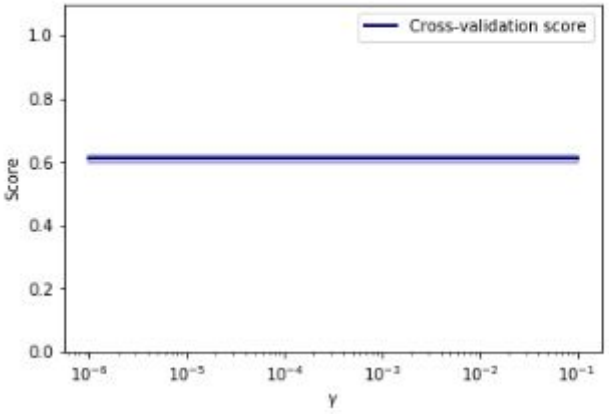


CROSS VALIDATION SCORE SIGMOID C=10 : 0.46849624060150374

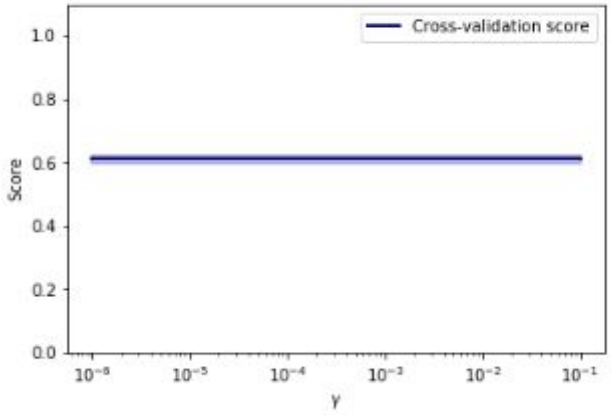


c. LINEAR

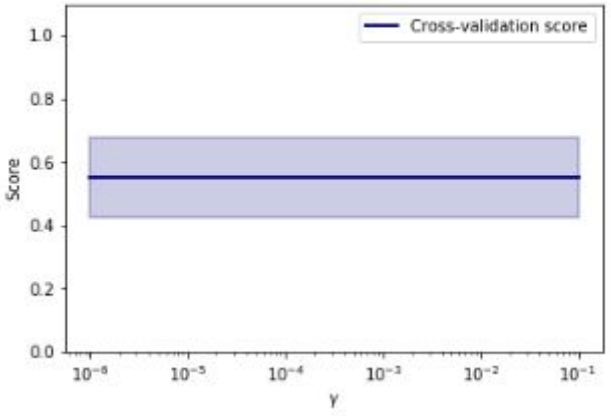
CROSS VALIDATION SCORE LINEAR C=0.01 : 0.610125313283208



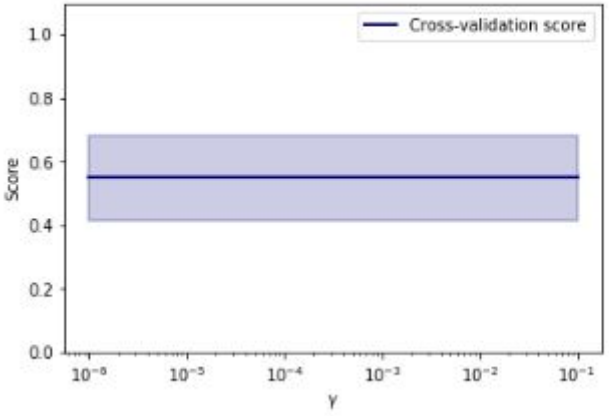
CROSS VALIDATION SCORE LINEAR C=0.1 : 0.610125313283208



CROSS VALIDATION SCORE LINEAR C=1 : 0.610125313283208



CROSS VALIDATION SCORE LINEAR C=10 : 0.46849624060150374



Naive-Bayes dengan data Visit-Nominal.csv

1. Inisialisasikan library yang diperlukan untuk dataset ini.

In [12]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statistics

from sklearn.naive_bayes import BernoulliNB

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
```

2. Masukan dataset Visit-Nominal.csv kedalam dataframe

In [13]:
```
df = pd.read_csv("\\Users\\aftermath\\Documents\\Machine Learning\\Visit-Nominal.csv",header=None,
skipinitialspace=True)
attrs = []
for attr in range(7):
    attrs.append(df.at[0,attr])

dfnew = pd.read_csv("\\Users\\aftermath\\Documents\\Machine Learning\\Visit-Nominal.csv",header=No
ne, skipinitialspace=True, skiprows = 1)
dfnew.columns = attrs

dfnew.head()
```

Out[13]:

	Home	Browsed	Searched	Prod_A	Prod_B	Prod_C	Visit_Again
0	yes	no	no	no	no	no	no
1	yes	yes	yes	no	no	no	no
2	yes	no	no	no	no	no	no
3	yes	yes	yes	yes	no	no	yes
4	yes	no	yes	yes	yes	no	yes

In [14]:
```
print(dfnew.describe())
```

```
        Home Browsed Searched Prod_A Prod_B Prod_C Visit_Again
count    100     100      100    100    100    100         100
unique     2       2        2      2      2      2           2
top      yes     yes       no    yes    yes     no          no
freq      60      72       57     53     55     55          61
```

3. Representasikan data 'yes' dan 'no' kedalam bentuk biner '1' dan '0'

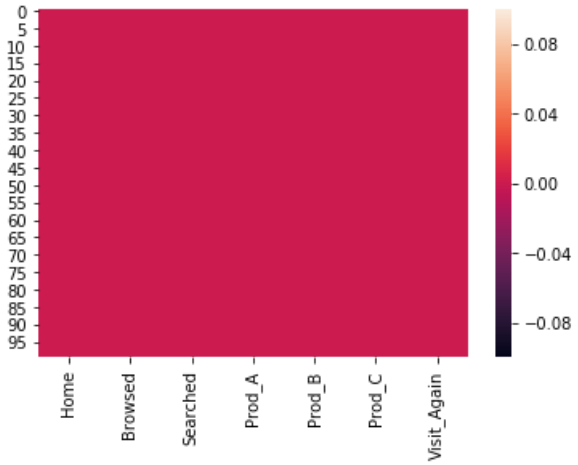
In [15]:
```
toBinary = lambda x: 1 if x=="yes" else 0
dfnew = dfnew.applymap(toBinary)
```

4. Cek apakah terdapat value fitur yang kosong

In [16]:
```
missing_values = dfnew.isnull()
missing_values

sns.heatmap(data = missing_values)
```

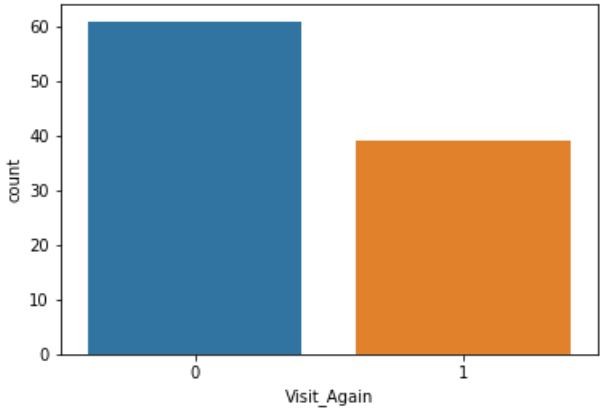
Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2184911fb70>
```



5. Lihat perbandingan jumlah class 0 dan 1

In [17]:
```
sns.countplot(x='Visit_Again', data=dfnew)
dfnew.Visit_Again.value_counts()
```

Out[17]:
```
0    61
1    39
Name: Visit_Again, dtype: int64
```



6. Siapkan data untuk training secara biasa dan training dengan cross validation

In [18]:
```
feature = attrs
feature.pop()
feature
```

Out[18]:
```
['Home', 'Browsed', 'Searched', 'Prod_A', 'Prod_B', 'Prod_C']
```

In [19]:
```
features = dfnew[feature]
label = dfnew['Visit_Again']

X_train, X_test, y_train, y_test = train_test_split(features, label, test_size=0.25, random_state=
101)
feature
```

Out[19]:
```
['Home', 'Browsed', 'Searched', 'Prod_A', 'Prod_B', 'Prod_C']
```

7. Train data secara biasa, didapatkan skor sebesar 0.64

In [20]:
```
naivebayesmodel = BernoulliNB()

naivebayesmodel.fit(X_train, y_train)
print("SCORE : ",naivebayesmodel.score(X_test, y_test))
```

```
SCORE :  0.56
```

8. Train data dengan cross-validation, didapatkan skor sebesar 0.58

In [21]:
```
naivebayesmodelkfold = BernoulliNB()

score =  cross_val_score(naivebayesmodelkfold, features, label, cv=5)
print("CROSS VALIDATION SCORE : ",statistics.mean(score))
```

```
CROSS VALIDATION SCORE :  0.5200751879699248
```

9. Melihat log probabilitas setiap fitur dari class 0 dan 1

In [22]:
```
class0_attrprob = naivebayesmodel.feature_log_prob_[0]
class1_attrprob = naivebayesmodel.feature_log_prob_[1]
print("probability of class 0 ")
for i in (range(len(naivebayesmodel.feature_log_prob_[0]))):
    print("feature ",i+1,": ",pow(2,class0_attrprob[i]))

print("\nprobability of class 1 ")
for i in (range(len(naivebayesmodel.feature_log_prob_[1]))):
    print("feature ",i+1,": ",pow(2,class1_attrprob[i]))
```

```
probability of class 0
feature  1 :  0.732575714272584
feature  2 :  0.7826070218144325
feature  3 :  0.5337704624000148
feature  4 :  0.6275952022872798
feature  5 :  0.7155617759223357
feature  6 :  0.6093515137984894

probability of class 1
feature  1 :  0.6711168360853097
feature  2 :  0.7954032230077729
feature  3 :  0.4465362457883447
feature  4 :  0.7712692781806878
feature  5 :  0.6711168360853097
feature  6 :  0.591444335183008
```

In [ ]:

Support Vector Machine dengan data Visit-Nominal.csv

1. Inisialisasikan library yang diperlukan untuk dataset ini.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statistics

from sklearn import svm
from sklearn.datasets import load_digits
from sklearn.model_selection import validation_curve
from sklearn.model_selection import cross_val_score
```

2. Masukan dataset Visit-Nominal.csv kedalam dataframe

In [2]:
```
df = pd.read_csv("\\Users\\aftermath\\Documents\\Machine Learning\\Visit-Nominal.csv",header=None,
skipinitialspace=True)
attrs = []
for attr in range(7):
    attrs.append(df.at[0,attr])

dfnew = pd.read_csv("\\Users\\aftermath\\Documents\\Machine Learning\\Visit-Nominal.csv",header=No
ne, skipinitialspace=True, skiprows = 1)
dfnew.columns = attrs

dfnew.head()
```

Out[2]:

	Home	Browsed	Searched	Prod_A	Prod_B	Prod_C	Visit_Again
0	yes	no	no	no	no	no	no
1	yes	yes	yes	no	no	no	no
2	yes	no	no	no	no	no	no
3	yes	yes	yes	yes	no	no	yes
4	yes	no	yes	yes	yes	no	yes

In [3]:
```
print(dfnew.describe())
```

```
        Home Browsed Searched Prod_A Prod_B Prod_C Visit_Again
count    100     100      100    100    100    100         100
unique     2       2        2      2      2      2           2
top      yes     yes       no    yes    yes     no          no
freq      60      72       57     53     55     55          61
```

3. Representasikan data 'yes' dan 'no' kedalam bentuk biner '1' dan '0'

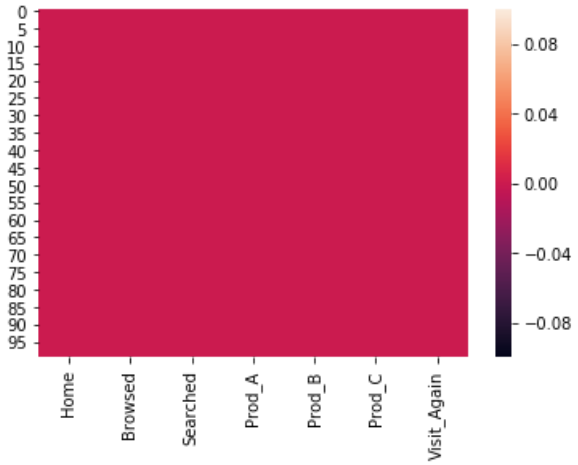
In [4]:
```
toBinary = lambda x: 1 if x=="yes" else 0
dfnew = dfnew.applymap(toBinary)
```

4. Cek apakah terdapat value fitur yang kosong

In [5]:
```
missing_values = dfnew.isnull()
missing_values

sns.heatmap(data = missing_values)
```

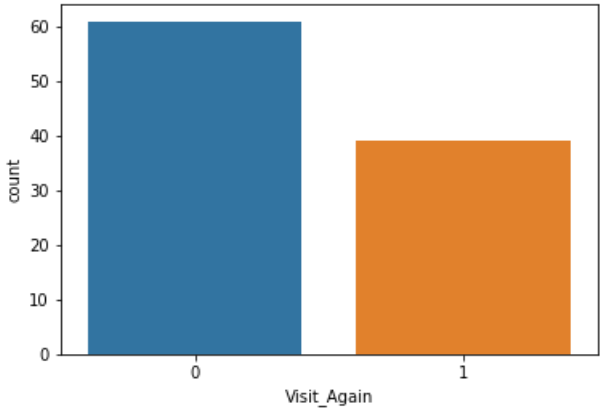
Out[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x263e84d4588>
```



5. Lihat perbandingan jumlah class 0 dan 1

In [6]:
```
sns.countplot(x='Visit_Again', data=dfnew)
dfnew.Visit_Again.value_counts()
```

Out[6]:
```
0    61
1    39
Name: Visit_Again, dtype: int64
```



6. Siapkan data untuk training secara biasa dan training dengan cross validation

In [7]:
```
feature = attrs
feature.pop()
feature
```

Out[7]:
```
['Home', 'Browsed', 'Searched', 'Prod_A', 'Prod_B', 'Prod_C']
```

In [8]:
```
features = dfnew[feature]
label = dfnew['Visit_Again']
```

7. Inisialisasi 12 model

In [9]:
```
#RBF C = 0.01
rbfmodel1 = svm.SVC(gamma='scale', C=0.01, kernel='rbf')
#RBF C = 0.1
rbfmodel2 = svm.SVC(gamma='scale', C=0.1, kernel='rbf')
#RBF C = 1
rbfmodel3 = svm.SVC(gamma='scale', C=1, kernel='rbf')
#RBF C = 10
rbfmodel4 = svm.SVC(gamma='scale', C=10, kernel='rbf')

#SIGMOID C = 0.01
sigmoidmodel1 = svm.SVC(gamma='scale', C=0.01, kernel='sigmoid')
#SIGMOID C = 0.1
sigmoidmodel2 = svm.SVC(gamma='scale', C=0.1, kernel='sigmoid')
#SIGMOID C = 1
sigmoidmodel3 = svm.SVC(gamma='scale', C=1, kernel='sigmoid')
#SIGMOID C = 10
sigmoidmodel4 = svm.SVC(gamma='scale', C=10, kernel='sigmoid')

#LINEAR C = 0.01
linearmodel1 = svm.SVC(gamma='scale', C=0.01, kernel='linear')
#LINEAR C = 0.1
linearmodel2 = svm.SVC(gamma='scale', C=0.1, kernel='linear')
#LINEAR C = 1
linearmodel3 = svm.SVC(gamma='scale', C=1, kernel='linear')
#LINEAR C = 10
linearmodel4 = svm.SVC(gamma='scale', C=10, kernel='linear')
```

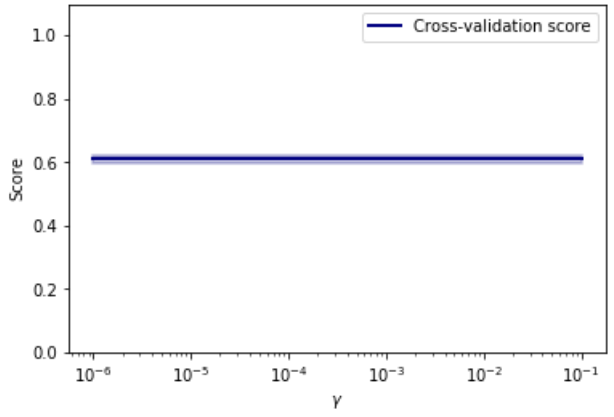
7.1 RBF Model

In [10]:
```
RBFscore1 =  cross_val_score(rbfmodel1, features, label, cv=5)
print("CROSS VALIDATION SCORE RBF C=0.01 : ",statistics.mean(RBFscore1))

param_range = np.logspace(-6, -1, 5)
train_scores, test_scores = validation_curve(
    rbfmodel1, features, label, param_name="gamma", param_range=param_range,
    cv=5, scoring="accuracy", n_jobs=1)
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)

plt.xlabel(r"$\gamma$")
plt.ylabel("Score")
plt.ylim(0.0, 1.1)
lw = 2
plt.semilogx(param_range, test_scores_mean, label="Cross-validation score",
             color="navy", lw=lw)
plt.fill_between(param_range, test_scores_mean - test_scores_std,
                 test_scores_mean + test_scores_std, alpha=0.2,
                 color="navy", lw=lw)
plt.legend(loc="best")
plt.show()
```

```
CROSS VALIDATION SCORE RBF C=0.01 :  0.610125313283208
```

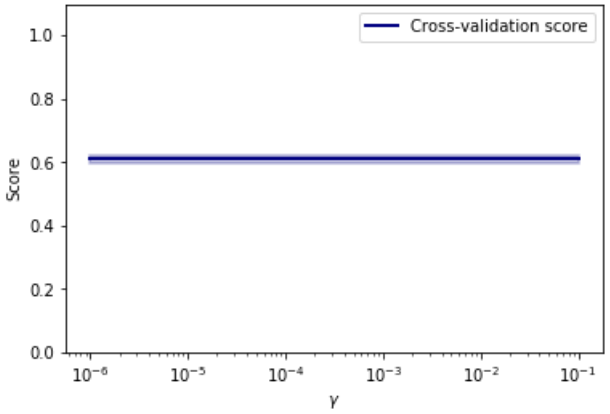


In [11]:
```
RBFscore2 =  cross_val_score(rbfmodel2, features, label, cv=5)
print("CROSS VALIDATION SCORE RBF C=0.1 : ",statistics.mean(RBFscore2))

train_scores, test_scores = validation_curve(
    rbfmodel2, features, label, param_name="gamma", param_range=param_range,
    cv=5, scoring="accuracy", n_jobs=1)
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)

plt.xlabel(r"$\gamma$")
plt.ylabel("Score")
plt.ylim(0.0, 1.1)
lw = 2
plt.semilogx(param_range, test_scores_mean, label="Cross-validation score",
             color="navy", lw=lw)
plt.fill_between(param_range, test_scores_mean - test_scores_std,
                 test_scores_mean + test_scores_std, alpha=0.2,
                 color="navy", lw=lw)
plt.legend(loc="best")
plt.show()
```

```
CROSS VALIDATION SCORE RBF C=0.1 :  0.610125313283208
```

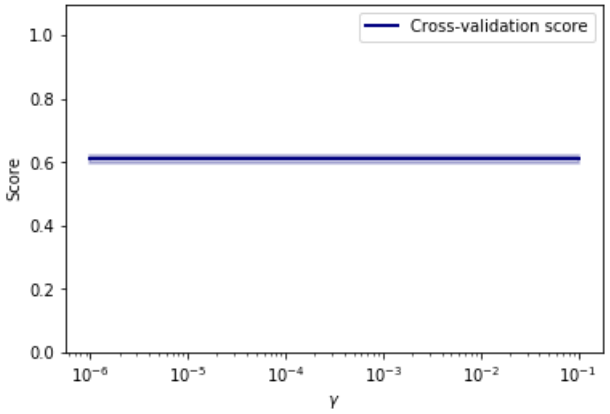


In [12]:
```
RBFscore3 =  cross_val_score(rbfmodel3, features, label, cv=5)
print("CROSS VALIDATION SCORE RBF C=1 : ",statistics.mean(RBFscore3))

train_scores, test_scores = validation_curve(
    rbfmodel3, features, label, param_name="gamma", param_range=param_range,
    cv=5, scoring="accuracy", n_jobs=1)
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)

plt.xlabel(r"$\gamma$")
plt.ylabel("Score")
plt.ylim(0.0, 1.1)
lw = 2
plt.semilogx(param_range, test_scores_mean, label="Cross-validation score",
             color="navy", lw=lw)
plt.fill_between(param_range, test_scores_mean - test_scores_std,
                 test_scores_mean + test_scores_std, alpha=0.2,
                 color="navy", lw=lw)
plt.legend(loc="best")
plt.show()
```

```
CROSS VALIDATION SCORE RBF C=1 :  0.5095989974937344
```

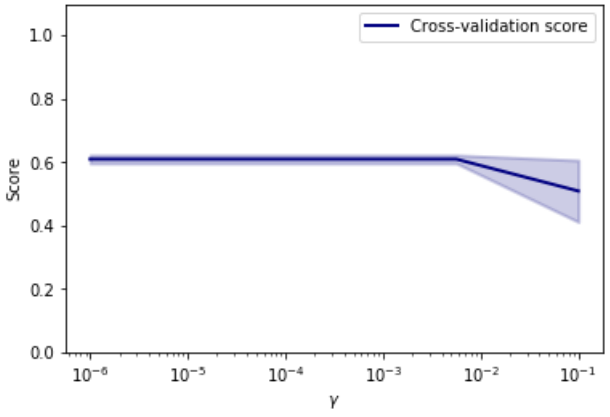


In [13]:
```
RBFscore4 =  cross_val_score(rbfmodel2, features, label, cv=5)
print("CROSS VALIDATION SCORE RBF C=10 : ",statistics.mean(RBFscore4))

train_scores, test_scores = validation_curve(
    rbfmodel4, features, label, param_name="gamma", param_range=param_range,
    cv=5, scoring="accuracy", n_jobs=1)
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)

plt.xlabel(r"$\gamma$")
plt.ylabel("Score")
plt.ylim(0.0, 1.1)
lw = 2
plt.semilogx(param_range, test_scores_mean, label="Cross-validation score",
             color="navy", lw=lw)
plt.fill_between(param_range, test_scores_mean - test_scores_std,
                 test_scores_mean + test_scores_std, alpha=0.2,
                 color="navy", lw=lw)
plt.legend(loc="best")
plt.show()
```

```
CROSS VALIDATION SCORE RBF C=10 :  0.610125313283208
```



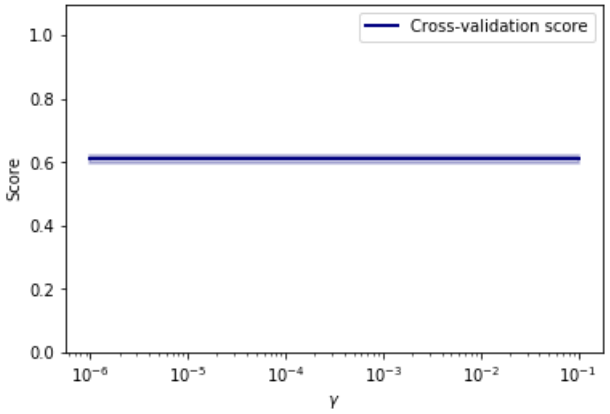
7.2 SIGMOID Model

In [14]:
```
sigmoidscore1 =  cross_val_score(sigmoidmodel1, features, label, cv=5)
print("CROSS VALIDATION SCORE SIGMOID C=0.01 : ",statistics.mean(sigmoidscore1))

train_scores, test_scores = validation_curve(
    sigmoidmodel1, features, label, param_name="gamma", param_range=param_range,
    cv=5, scoring="accuracy", n_jobs=1)
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)

plt.xlabel(r"$\gamma$")
plt.ylabel("Score")
plt.ylim(0.0, 1.1)
lw = 2
plt.semilogx(param_range, test_scores_mean, label="Cross-validation score",
             color="navy", lw=lw)
plt.fill_between(param_range, test_scores_mean - test_scores_std,
                 test_scores_mean + test_scores_std, alpha=0.2,
                 color="navy", lw=lw)
plt.legend(loc="best")
plt.show()
```

```
CROSS VALIDATION SCORE SIGMOID C=0.01 :  0.610125313283208
```

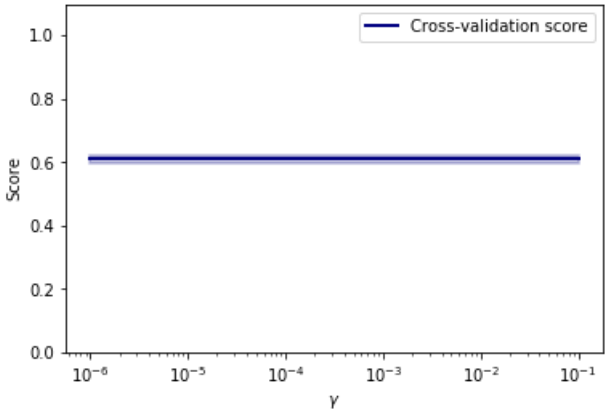


In [15]:
```
sigmoidscore2 =  cross_val_score(sigmoidmodel2, features, label, cv=5)
print("CROSS VALIDATION SCORE SIGMOID C=0.1 : ",statistics.mean(sigmoidscore2))

train_scores, test_scores = validation_curve(
    sigmoidmodel2, features, label, param_name="gamma", param_range=param_range,
    cv=5, scoring="accuracy", n_jobs=1)
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)

plt.xlabel(r"$\gamma$")
plt.ylabel("Score")
plt.ylim(0.0, 1.1)
lw = 2
plt.semilogx(param_range, test_scores_mean, label="Cross-validation score",
             color="navy", lw=lw)
plt.fill_between(param_range, test_scores_mean - test_scores_std,
                 test_scores_mean + test_scores_std, alpha=0.2,
                 color="navy", lw=lw)
plt.legend(loc="best")
plt.show()
```

```
CROSS VALIDATION SCORE SIGMOID C=0.1 :  0.610125313283208
```

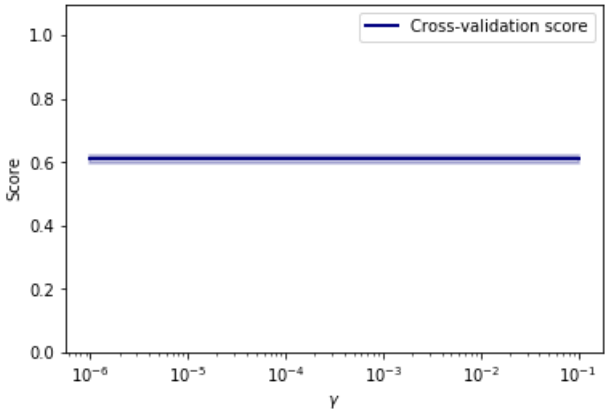


In [16]:
```
sigmoidscore3 =  cross_val_score(sigmoidmodel3, features, label, cv=5)
print("CROSS VALIDATION SCORE SIGMOID C=1 : ",statistics.mean(sigmoidscore3))

train_scores, test_scores = validation_curve(
    sigmoidmodel3, features, label, param_name="gamma", param_range=param_range,
    cv=5, scoring="accuracy", n_jobs=1)
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)

plt.xlabel(r"$\gamma$")
plt.ylabel("Score")
plt.ylim(0.0, 1.1)
lw = 2
plt.semilogx(param_range, test_scores_mean, label="Cross-validation score",
             color="navy", lw=lw)
plt.fill_between(param_range, test_scores_mean - test_scores_std,
                 test_scores_mean + test_scores_std, alpha=0.2,
                 color="navy", lw=lw)
plt.legend(loc="best")
plt.show()
```

```
CROSS VALIDATION SCORE SIGMOID C=1 :  0.640125313283208
```

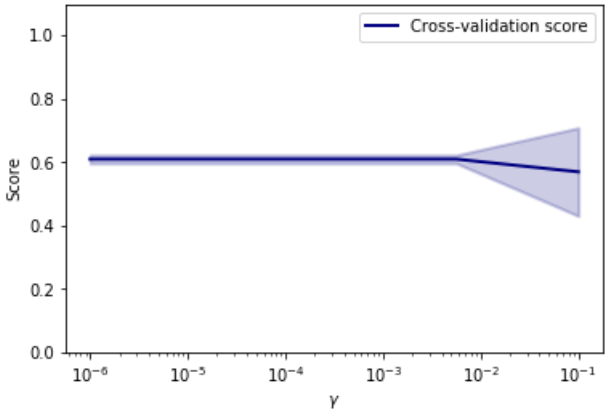


In [17]:
```
sigmoidscore4 =  cross_val_score(sigmoidmodel4, features, label, cv=5)
print("CROSS VALIDATION SCORE SIGMOID C=10 : ",statistics.mean(sigmoidscore4))

train_scores, test_scores = validation_curve(
    sigmoidmodel4, features, label, param_name="gamma", param_range=param_range,
    cv=5, scoring="accuracy", n_jobs=1)
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)

plt.xlabel(r"$\gamma$")
plt.ylabel("Score")
plt.ylim(0.0, 1.1)
lw = 2
plt.semilogx(param_range, test_scores_mean, label="Cross-validation score",
             color="navy", lw=lw)
plt.fill_between(param_range, test_scores_mean - test_scores_std,
                 test_scores_mean + test_scores_std, alpha=0.2,
                 color="navy", lw=lw)
plt.legend(loc="best")
plt.show()
```

```
CROSS VALIDATION SCORE SIGMOID C=10 :  0.46849624060150374
```



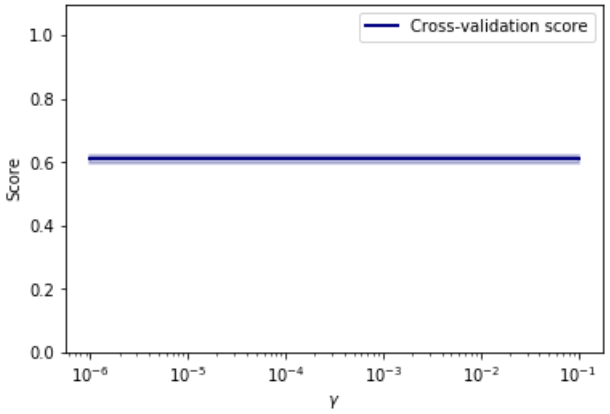
7.3 LINEAR Model

In [18]:
```
linearscore1 =  cross_val_score(sigmoidmodel1, features, label, cv=5)
print("CROSS VALIDATION SCORE LINEAR C=0.01 : ",statistics.mean(linearscore1))


train_scores, test_scores = validation_curve(
    linearmodel1, features, label, param_name="gamma", param_range=param_range,
    cv=5, scoring="accuracy", n_jobs=1)
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)

plt.xlabel(r"$\gamma$")
plt.ylabel("Score")
plt.ylim(0.0, 1.1)
lw = 2
plt.semilogx(param_range, test_scores_mean, label="Cross-validation score",
             color="navy", lw=lw)
plt.fill_between(param_range, test_scores_mean - test_scores_std,
                 test_scores_mean + test_scores_std, alpha=0.2,
                 color="navy", lw=lw)
plt.legend(loc="best")
plt.show()
```

```
CROSS VALIDATION SCORE LINEAR C=0.01 :  0.610125313283208
```

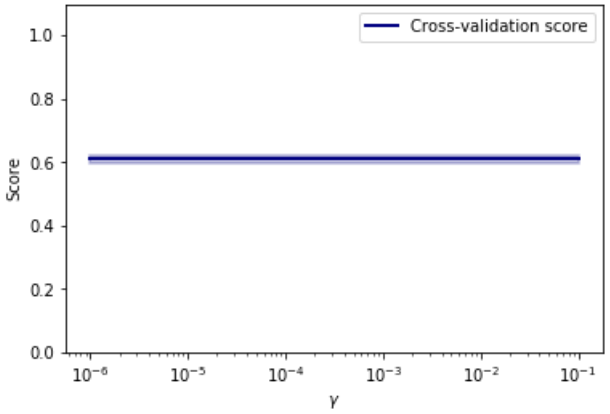


In [19]:
```
linearscore2 =  cross_val_score(sigmoidmodel2, features, label, cv=5)
print("CROSS VALIDATION SCORE LINEAR C=0.1 : ",statistics.mean(linearscore2))

train_scores, test_scores = validation_curve(
    linearmodel2, features, label, param_name="gamma", param_range=param_range,
    cv=5, scoring="accuracy", n_jobs=1)
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)

plt.xlabel(r"$\gamma$")
plt.ylabel("Score")
plt.ylim(0.0, 1.1)
lw = 2
plt.semilogx(param_range, test_scores_mean, label="Cross-validation score",
             color="navy", lw=lw)
plt.fill_between(param_range, test_scores_mean - test_scores_std,
                 test_scores_mean + test_scores_std, alpha=0.2,
                 color="navy", lw=lw)
plt.legend(loc="best")
plt.show()
```

```
CROSS VALIDATION SCORE LINEAR C=0.1 :  0.610125313283208
```

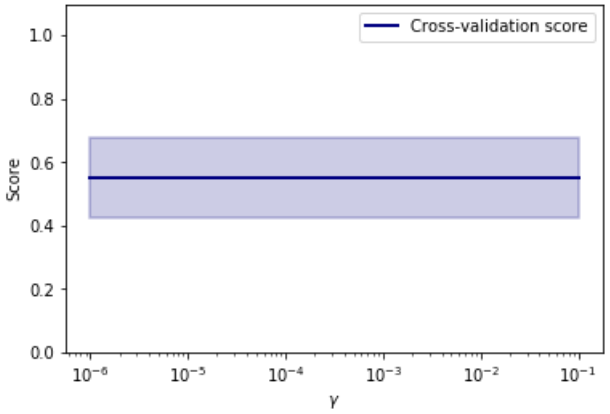


In [20]:
```
linearscore3 =  cross_val_score(sigmoidmodel1, features, label, cv=5)
print("CROSS VALIDATION SCORE LINEAR C=1 : ",statistics.mean(linearscore3))

train_scores, test_scores = validation_curve(
    linearmodel3, features, label, param_name="gamma", param_range=param_range,
    cv=5, scoring="accuracy", n_jobs=1)
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)

plt.xlabel(r"$\gamma$")
plt.ylabel("Score")
plt.ylim(0.0, 1.1)
lw = 2
plt.semilogx(param_range, test_scores_mean, label="Cross-validation score",
             color="navy", lw=lw)
plt.fill_between(param_range, test_scores_mean - test_scores_std,
                 test_scores_mean + test_scores_std, alpha=0.2,
                 color="navy", lw=lw)
plt.legend(loc="best")
plt.show()
```

```
CROSS VALIDATION SCORE LINEAR C=1 :  0.610125313283208
```



In [21]:
```
linearscore4 =  cross_val_score(sigmoidmodel4, features, label, cv=5)
print("CROSS VALIDATION SCORE LINEAR C=10 : ",statistics.mean(linearscore4))

train_scores, test_scores = validation_curve(
    linearmodel4, features, label, param_name="gamma", param_range=param_range,
    cv=5, scoring="accuracy", n_jobs=1)
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)

plt.xlabel(r"$\gamma$")
plt.ylabel("Score")
plt.ylim(0.0, 1.1)
lw = 2
plt.semilogx(param_range, test_scores_mean, label="Cross-validation score",
             color="navy", lw=lw)
plt.fill_between(param_range, test_scores_mean - test_scores_std,
                 test_scores_mean + test_scores_std, alpha=0.2,
                 color="navy", lw=lw)
plt.legend(loc="best")
plt.show()
```

```
CROSS VALIDATION SCORE LINEAR C=10 :  0.46849624060150374
```