4105 Hw 4, Owen Evans, 801206774

In [1]:
```
import numpy as np
import pandas as pd
import nbconvert
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import datasets
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.decomposition import PCA
from sklearn import preprocessing
from sklearn.naive_bayes import GaussianNB
import warnings
warnings.filterwarnings('ignore')
from sklearn.datasets import load_breast_cancer


from sklearn import model_selection
from sklearn.preprocessing import StandardScaler, MinMaxScaler

from sklearn import metrics
from sklearn.svm import SVC, SVR
```

In [2]:
```
cancer = load_breast_cancer()
Data=cancer.data
Data.shape
```

Out[2]:
```
(569, 30)
```

In [3]:
```
x=pd.DataFrame(Data)
x.head()
```

Out[3]:

	0	1	2	3	4	5	6	7	8	9	...	20	2
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	25.38	17.3
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	24.99	23.4
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	23.57	25.5
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	14.91	26.5
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	22.54	16.6

5 rows × 30 columns

In [4]:
```
Target= cancer.target
y=pd.DataFrame(Target)
y.head()
```

Out[4]:

	0
0	0
1	0
2	0
3	0
4	0

In [35]:
```
labels = np.reshape(y,(569,1))
final_breast_data = np.concatenate([Data,labels],axis=1)
final_breast_data.shape
breast_dataset = pd.DataFrame(final_breast_data)
features = cancer.feature_names
features_labels = np.append(features,'label')
breast_dataset.columns = features_labels
breast_dataset.head()
```

Out[35]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	d
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	

5 rows × 31 columns

In [6]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)
```

In [116]:
```
C=[0.001,0.1,1,0.01]
bestC=0
best_acc=0
ValAcc=0
TrainAcc=0


for c in C:
    model = SVC(kernel='linear',C=c)
    model.fit(x_train,y_train)
    train_acc = model.score(x_train,y_train)
    val_acc = model.score(x_test,y_test)
    print("C:",c)
    print("Validation accuracy:",val_acc)
    print("")
    if(val_acc>best_acc):
        best_acc = val_acc
        bestC = c
        ValAcc = val_acc
        TrainAcc = train_acc
print('Best C:',bestC)
print('Training accuracy:',TrainAcc)
print('Validation accuracy:',ValAcc)
```

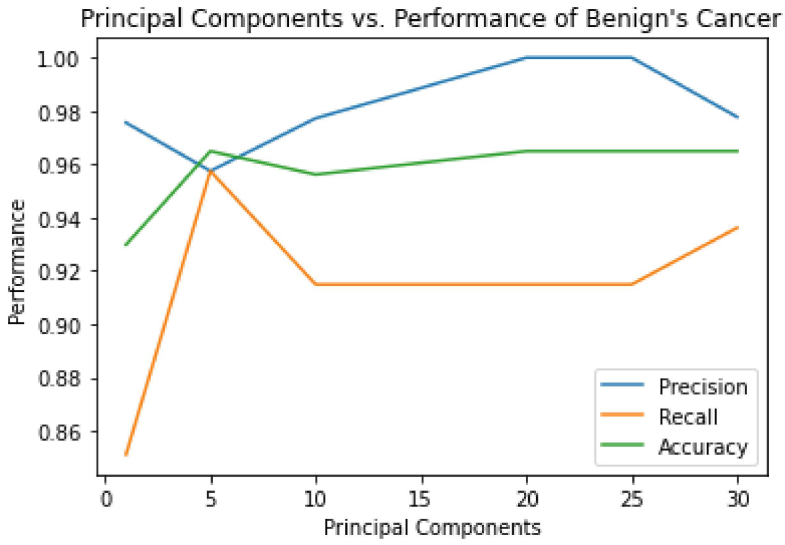
```
C: 0.001
Validation accuracy: 0.9385964912280702

C: 0.1
Validation accuracy: 0.9473684210526315

C: 1
Validation accuracy: 0.956140350877193

C: 0.01
Validation accuracy: 0.9473684210526315

Best C: 1
Training accuracy: 0.9626373626373627
Validation accuracy: 0.956140350877193
```

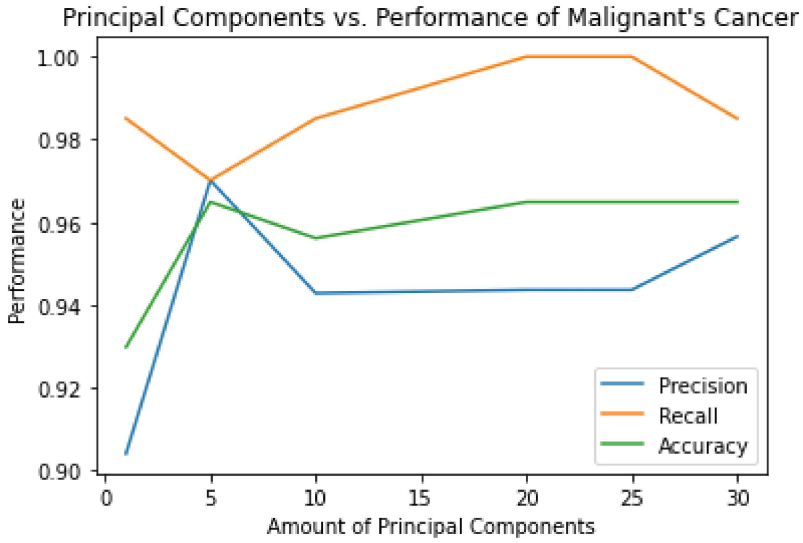
In [171]:
```
Recall_0 = []
Recall_1 = []
Prec_0 = []
Prec_1 = []
Acc = []
best_acc = 0

K=[1,5,10,20,25,30]

for k in K:

    pca = PCA(n_components=k)
    comps = pca.fit_transform(x)
    X = pd.DataFrame(data = comps)

    x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, ra

    sc_X = StandardScaler()
    x_train = sc_X.fit_transform(x_train)
    x_test = sc_X.fit_transform(x_test)

    svc = SVC(kernel='linear', C = bestC)
    model = svc.fit(x_train, y_train)
    pred = model.predict(x_test)

    train_acc = svc.score(x_train,y_train)
    val_acc = svc.score(x_test,y_test)

    print('K:',k,'Train',train_acc,'Validation',val_acc)

    predicted = svc.predict(x_test)
    matrix = confusion_matrix(y_test, predicted)

    report = classification_report(y_test, pred, output_dict=True)
    Data = pd.DataFrame(report)

    Recall_0.append(Data.values[1,0])
    Recall_1.append(Data.values[1,1])
    Prec_0.append(Data.values[0,0])
    Prec_1.append(Data.values[0,1])
    Acc.append(Data.values[0,2])


    if(val_acc>best_acc):
        best_acc = val_acc
        best_k = k
        TrainAcc=train_acc

print('best k:',best_k)
preds = svc.predict(x_test_pca)
cr = classification_report(y_test,pred)
print(cr)
print('Training accuracy:',TrainAcc)
print('Validation accuracy:',best_acc)
print('')
print('matrix','\n',matrix)
```

```
K: 1 Train 0.9076923076923077 Validation 0.9298245614035088
K: 5 Train 0.9582417582417583 Validation 0.9649122807017544
K: 10 Train 0.9648351648351648 Validation 0.956140350877193
K: 20 Train 0.9846153846153847 Validation 0.9649122807017544
K: 25 Train 0.9868131868131869 Validation 0.9649122807017544
K: 30 Train 0.9868131868131869 Validation 0.9649122807017544
best k: 5
              precision    recall  f1-score   support

           0       0.98      0.94      0.96        47
           1       0.96      0.99      0.97        67

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114

Training accuracy: 0.9582417582417583
Validation accuracy: 0.9649122807017544

matrix
 [[44  3]
 [ 1 66]]
```

In [172]:
```
print('Linear :')
plt.plot(K, Prec_0, label = 'Precision')
plt.plot(K, Recall_0, label = 'Recall')
plt.plot(K, Acc, label = 'Accuracy')
plt.title("Principal Components vs. Performance of Benign's Cancer")
plt.xlabel("Principal Components")
plt.ylabel("Performance")
plt.legend()
plt.show()
plt.plot(K, Prec_1, label = "Precision")
plt.plot(K, Recall_1, label = "Recall")
plt.plot(K, Acc, label = "Accuracy")
plt.title("Principal Components vs. Performance of Malignant's Cancer")
plt.xlabel("Amount of Principal Components")
plt.ylabel("Performance")
plt.legend()
plt.show()
```

```
Linear :
```



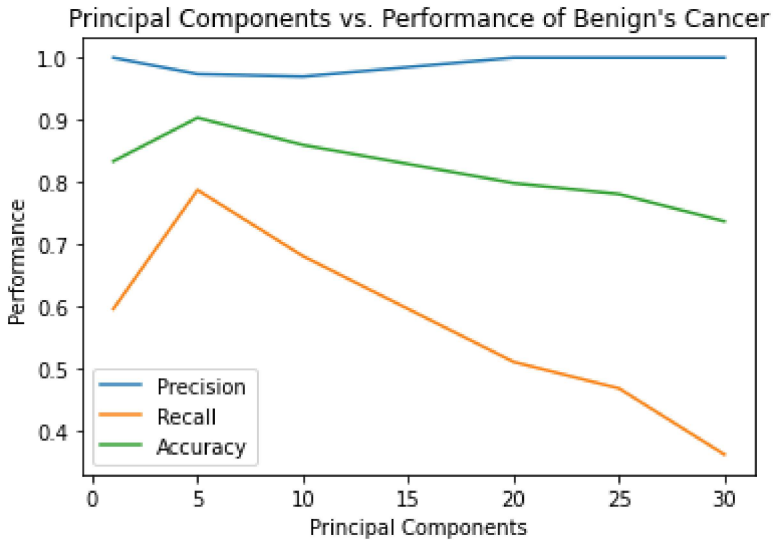


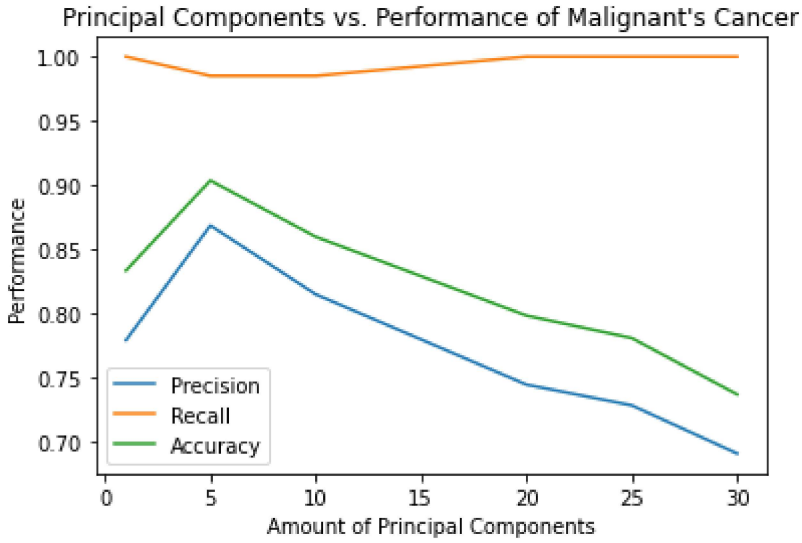
In [173]:
```
Recall_0 = []
Recall_1 = []
Prec_0 = []
Prec_1 = []
Acc = []
best_acc = 0

K=[1,5,10,20,25,30]

for k in K:

    pca = PCA(n_components=k)
    comps = pca.fit_transform(x)
    X = pd.DataFrame(data = comps)

    x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, ra

    sc_X = StandardScaler()
    x_train = sc_X.fit_transform(x_train)
    x_test = sc_X.fit_transform(x_test)

    svc = SVC(kernel='poly', C = bestC)
    model = svc.fit(x_train, y_train)
    pred = model.predict(x_test)

    train_acc = svc.score(x_train,y_train)
    val_acc = svc.score(x_test,y_test)

    print('K:',k,'Train',train_acc,'Validation',val_acc)

    predicted = svc.predict(x_test)
    matrix = confusion_matrix(y_test, predicted)

    report = classification_report(y_test, pred, output_dict=True)
    Data = pd.DataFrame(report)

    Recall_0.append(Data.values[1,0])
    Recall_1.append(Data.values[1,1])
    Prec_0.append(Data.values[0,0])
    Prec_1.append(Data.values[0,1])
    Acc.append(Data.values[0,2])


    if(val_acc>best_acc):
        best_acc = val_acc
        best_k = k
        TrainAcc=train_acc

print('best k:',best_k)
preds = svc.predict(x_test_pca)
cr = classification_report(y_test,pred)
print(cr)
print('Training accuracy:',TrainAcc)
print('Validation accuracy:',best_acc)
print('')
print('matrix','\n',matrix)
```

```
K: 1 Train 0.8593406593406593 Validation 0.8333333333333334
K: 5 Train 0.8879120879120879 Validation 0.9035087719298246
K: 10 Train 0.9010989010989011 Validation 0.8596491228070176
K: 20 Train 0.8879120879120879 Validation 0.7982456140350878
K: 25 Train 0.8901098901098901 Validation 0.7807017543859649
K: 30 Train 0.8703296703296703 Validation 0.7368421052631579
best k: 5
              precision    recall  f1-score   support

           0       1.00      0.36      0.53        47
           1       0.69      1.00      0.82        67

    accuracy                           0.74       114
   macro avg       0.85      0.68      0.67       114
weighted avg       0.82      0.74      0.70       114

Training accuracy: 0.8879120879120879
Validation accuracy: 0.9035087719298246

matrix
 [[17 30]
 [ 0 67]]
```

In [174]:
```
print("Poly :")
plt.plot(K, Prec_0, label = "Precision")
plt.plot(K, Recall_0, label = "Recall")
plt.plot(K, Acc, label = "Accuracy")
plt.title("Principal Components vs. Performance of Benign's Cancer")
plt.xlabel("Principal Components")
plt.ylabel("Performance")
plt.legend()
plt.show()
plt.plot(K, Prec_1, label = "Precision")
plt.plot(K, Recall_1, label = "Recall")
plt.plot(K, Acc, label = "Accuracy")
plt.title("Principal Components vs. Performance of Malignant's Cancer")
plt.xlabel("Amount of Principal Components")
plt.ylabel("Performance")
plt.legend()
plt.show()
```

```
Poly :
```



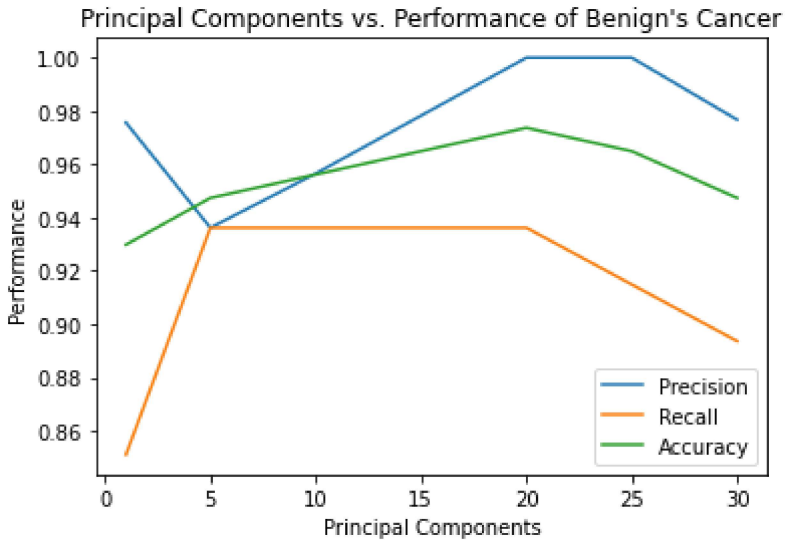


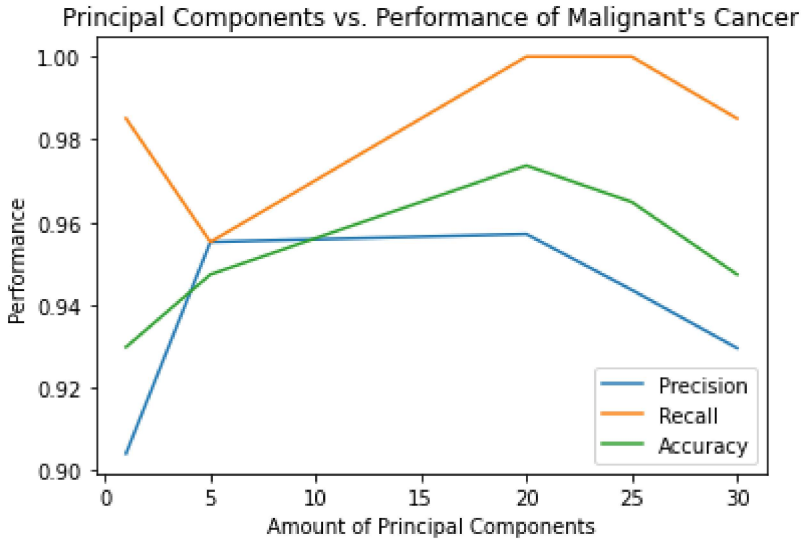
In [177]:
```
Recall_0 = []
Recall_1 = []
Prec_0 = []
Prec_1 = []
Acc = []
best_acc = 0

K=[1,5,10,20,25,30]

for k in K:

    pca = PCA(n_components=k)
    comps = pca.fit_transform(x)
    X = pd.DataFrame(data = comps)

    x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, ra

    sc_X = StandardScaler()
    x_train = sc_X.fit_transform(x_train)
    x_test = sc_X.fit_transform(x_test)

    svc = SVC(kernel='rbf', C = bestC)
    model = svc.fit(x_train, y_train)
    pred = model.predict(x_test)

    train_acc = svc.score(x_train,y_train)
    val_acc = svc.score(x_test,y_test)

    print('K:',k,'Train',train_acc,'Validation',val_acc)

    predicted = svc.predict(x_test)
    matrix = confusion_matrix(y_test, predicted)

    report = classification_report(y_test, pred, output_dict=True)
    Data = pd.DataFrame(report)

    Recall_0.append(Data.values[1,0])
    Recall_1.append(Data.values[1,1])
    Prec_0.append(Data.values[0,0])
    Prec_1.append(Data.values[0,1])
    Acc.append(Data.values[0,2])


    if(val_acc>best_acc):
        best_acc = val_acc
        best_k = k
        TrainAcc=train_acc

print('best k:',best_k)
preds = svc.predict(x_test_pca)
cr = classification_report(y_test,pred)
print(cr)
print('Training accuracy:',TrainAcc)
print('Validation accuracy:',best_acc)
print('')
print('matrix','\n',matrix)
```

```
K: 1 Train 0.9032967032967033 Validation 0.9298245614035088
K: 5 Train 0.9692307692307692 Validation 0.9473684210526315
K: 10 Train 0.9802197802197802 Validation 0.956140350877193
K: 20 Train 0.978021978021978 Validation 0.9736842105263158
K: 25 Train 0.9824175824175824 Validation 0.9649122807017544
K: 30 Train 0.9824175824175824 Validation 0.9473684210526315
best k: 20
              precision    recall  f1-score   support

           0       0.98      0.89      0.93        47
           1       0.93      0.99      0.96        67

    accuracy                           0.95       114
   macro avg       0.95      0.94      0.94       114
weighted avg       0.95      0.95      0.95       114

Training accuracy: 0.978021978021978
Validation accuracy: 0.9736842105263158

matrix
 [[42  5]
 [ 1 66]]
```

In [178]:
```
print("rbf :")
plt.plot(K, Prec_0, label = "Precision")
plt.plot(K, Recall_0, label = "Recall")
plt.plot(K, Acc, label = "Accuracy")
plt.title("Principal Components vs. Performance of Benign's Cancer")
plt.xlabel("Principal Components")
plt.ylabel("Performance")
plt.legend()
plt.show()
plt.plot(K, Prec_1, label = "Precision")
plt.plot(K, Recall_1, label = "Recall")
plt.plot(K, Acc, label = "Accuracy")
plt.title("Principal Components vs. Performance of Malignant's Cancer")
plt.xlabel("Amount of Principal Components")
plt.ylabel("Performance")
plt.legend()
plt.show()
```

```
rbf :
```





Problem 2

In [189]:
```
#get data
house = pd.read_csv ('https://raw.githubusercontent.com/Owen3vans/4105_HW1/main/H
house.head()
```

Out[189]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheati
0	13300000	7420	4	2	3	yes	no	no	
1	12250000	8960	4	4	4	yes	no	no	
2	12250000	9960	3	2	2	yes	no	yes	
3	12215000	7500	4	2	2	yes	no	yes	
4	11410000	7420	4	1	2	yes	yes	yes	

In [190]:
```
varlist= ['mainroad','guestroom','basement','hotwaterheating','airconditioning',
def bianary(x):
    return x.map({'yes':1, 'no':0})
house[varlist] = house[varlist].apply(bianary)
house.head()
```

Out[190]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheati
0	13300000	7420	4	2	3	1	0	0	
1	12250000	8960	4	4	4	1	0	0	
2	12250000	9960	3	2	2	1	0	1	
3	12215000	7500	4	2	2	1	0	1	
4	11410000	7420	4	1	2	1	1	1	

In [217]:
```
cl = ['area', 'bedrooms', 'bathrooms', 'stories', 'mainroad', 'guestroom', 'basem
        'airconditioning', 'parking', 'prefarea']
x = house[cl]
y = house['price']
y.head()
```

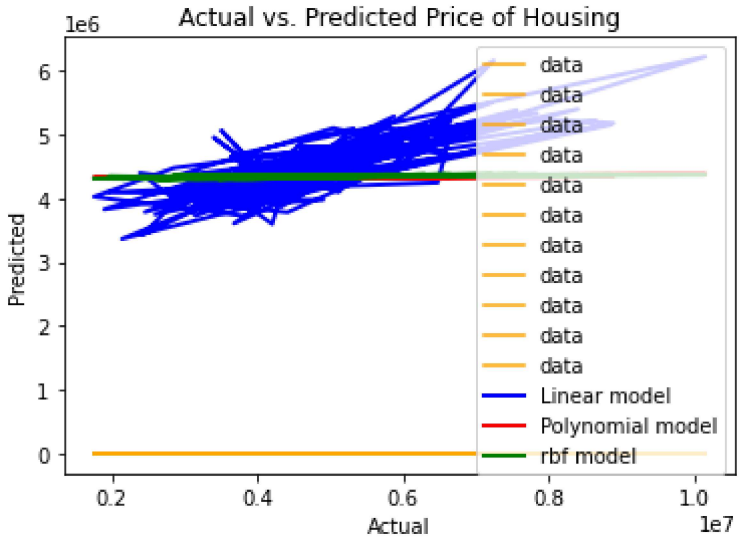
Out[217]:
```
0    13300000
1    12250000
2    12250000
3    12215000
4    11410000
Name: price, dtype: int64
```

In [284]:
```
sc = StandardScaler()
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
print(x_test)
```

```
[[-0.32522392  0.07567703 -0.55240601 ...  1.39497166 -0.67313865
  -0.57381904]
 [-1.23361405  0.07567703 -0.55240601 ... -0.71686044 -0.67313865
   1.74270968]
 [ 0.48601483 -1.2991223  -0.55240601 ...  1.39497166  0.66089976
  -0.57381904]
 ...
 [ 0.59204299  0.07567703 -0.55240601 ...  1.39497166 -0.67313865
   1.74270968]
 [-0.70988424  1.45047635 -0.55240601 ...  1.39497166 -0.67313865
  -0.57381904]
 [-0.19700381  0.07567703 -0.55240601 ...  1.39497166 -0.67313865
  -0.57381904]]
```

In [257]:
```
svr_linear=SVR(kernel='linear',C=1e3)
svr_poly = SVR(kernel='poly', C=1e3, degree=2)
svr_rbf = SVR(kernel = 'rbf', C=1e3, gamma=0.1)

y_lin = svr_linear.fit(x_train,y_train).predict(x_test)
y_poly= svr_poly.fit(x_train,y_train).predict(x_test)
y_rbf = svr_rbf.fit(x_train,y_train).predict(x_test)
```

In [258]:
```
plt.plot(y_test, x_test, color = 'orange', label= 'data')
plt.plot(y_test, y_lin,  color = 'blue',lw=2,label='Linear model')
plt.plot(y_test, y_poly, color = 'red', lw=2,label='Polynomial model')
plt.plot(y_test, y_rbf,  color = 'green',lw=2,label='rbf model')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs. Predicted Price of Housing')
plt.legend()
plt.show()
```



In [314]:
```
from sklearn.metrics import r2_score

linAcc=[]
bestlinAcc=0
bestlinK=0

K= range(1,len(cl)+1)
for k in K:
    pca = PCA(n_components = k)
    comp = pca.fit_transform(x)
    X_x = pd.DataFrame(data = comp)

    x_train, x_test, y_train, y_test = train_test_split(X_x, y, test_size = 0.2,

    sc_X = StandardScaler()
    x_train = sc_X.fit_transform(x_train)
    x_test = sc_X.fit_transform(x_test)

    svr = SVR(kernel='linear', C = 1e3)
    model = svr.fit(x_train, y_train)
    pred = model.predict(x_test)
    val_acc = (r2_score(y_test, pred))
    LinAcc.append(r2_score(y_test, pred))

    if(val_acc>bestlinAcc):
        bestlinAcc = val_acc
        bestlinK = k

print('Best accuracy is:',bestAcc)
print('with',bestrbfK,'components')
```

```
Best accuracy is: 0.37058338056157925
with 1 components
```

In [313]:
```
from sklearn.metrics import r2_score

polyAcc=[]
bestpolyAcc=0
bestpolyK=0

K= range(1,len(cl)+1)
for k in K:
    pca = PCA(n_components = k)
    comp = pca.fit_transform(x)
    X_x = pd.DataFrame(data = comp)

    x_train, x_test, y_train, y_test = train_test_split(X_x, y, test_size = 0.2,

    sc_X = StandardScaler()
    x_train = sc_X.fit_transform(x_train)
    x_test = sc_X.fit_transform(x_test)

    svr = SVR(kernel='poly', C = 1e3)
    model = svr.fit(x_train, y_train)
    pred = model.predict(x_test)
    val_acc = (r2_score(y_test, pred))
    polyAcc.append(r2_score(y_test, pred))

    if(val_acc>bestrbfAcc):
        bestpolyAcc = val_acc
        bestpolyK = k

print('Best accuracy is:',bestpolyAcc)
print('with',bestpolyK,'components')
```

```
Best accuracy is: 0.0021282060743015307
with 6 components
```

In [312]:
```
from sklearn.metrics import r2_score

rbfAcc=[]
bestrbfAcc=0
bestrbfK=0

K= range(1,len(cl)+1)
for k in K:
    pca = PCA(n_components = k)
    comp = pca.fit_transform(x)
    X_x = pd.DataFrame(data = comp)

    x_train, x_test, y_train, y_test = train_test_split(X_x, y, test_size = 0.2,

    sc_X = StandardScaler()
    x_train = sc_X.fit_transform(x_train)
    x_test = sc_X.fit_transform(x_test)

    svr = SVR(kernel='rbf', C = 1e3)
    model = svr.fit(x_train, y_train)
    pred = model.predict(x_test)
    val_acc = (r2_score(y_test, pred))
    rbfAcc.append(r2_score(y_test, pred))

    if(val_acc>bestrbfAcc):
        bestrbfAcc = val_acc
        bestrbfK = k

print('Best accuracy is:',bestrbfAcc)
print('with',bestrbfK,'components')
```

```
Best accuracy is: 0.0017276461124670073
with 1 components
```

In [ ]:
```

```