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

from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.decomposition import IncrementalPCA
from sklearn.metrics import classification_report
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_recall_curve
from sklearn import metrics
```

In [2]:
```
dataset = load_breast_cancer()

X = dataset.data

Y = dataset.target
```

In [3]:
```
X_train, X_test, y_train, y_test = train_test_split(X, Y, train_size = 0.8, test_size = 0.2, random_state = 0)

sc = StandardScaler()

Xsc_train = sc.fit_transform(X_train)
Xsc_test = sc.transform(X_test)
```

In [4]:
```
#sns.scatterplot(x=Xsc_train[:,0],y=Xsc_train[:,1],hue=y_train)
```

In [5]:
```
model = PCA()

model.fit(Xsc_train)

#explained_variance = np.cumsum(model.explained_variance_ratio_)
#plt.plot(explained_variance)
```

Out[5]:
```
PCA()
```

In [32]:
```
PCA_train = IncrementalPCA(n_components = 15)
df_train = PCA_train.fit_transform(Xsc_train)
df_test = PCA_train.transform(Xsc_test)
```

In [33]:
```
svc_lin = SVC(kernel='linear', C=100)
svc_rbf = SVC(kernel='rbf', C=100)
svc_ply = SVC(kernel='poly', C=100)

svc_lin = svc_lin.fit(df_train, y_train)
svc_rbf = svc_rbf.fit(df_train, y_train)
svc_ply = svc_ply.fit(df_train, y_train)
```

In [34]:
```
predicted_lin = svc_lin.predict(df_test)
predicted_rbf = svc_rbf.predict(df_test)
predicted_ply = svc_ply.predict(df_test)

print('Linear:',classification_report(y_test,predicted_lin, digits=4))
print('RBF:',classification_report(y_test,predicted_rbf, digits=4))
print('Poly:',classification_report(y_test,predicted_ply, digits=4))
```

```
Linear:               precision    recall  f1-score   support

           0     0.9375    0.9574    0.9474        47
           1     0.9697    0.9552    0.9624        67

    accuracy                         0.9561       114
   macro avg     0.9536    0.9563    0.9549       114
weighted avg     0.9564    0.9561    0.9562       114

RBF:                precision    recall  f1-score   support

           0     0.9038    1.0000    0.9495        47
           1     1.0000    0.9254    0.9612        67

    accuracy                         0.9561       114
   macro avg     0.9519    0.9627    0.9554       114
weighted avg     0.9604    0.9561    0.9564       114

Poly:                precision    recall  f1-score   support

           0     0.9783    0.9574    0.9677        47
           1     0.9706    0.9851    0.9778        67

    accuracy                         0.9737       114
   macro avg     0.9744    0.9713    0.9728       114
weighted avg     0.9738    0.9737    0.9736       114
```

In [35]:
```
print('Accuracy:',accuracy_score(y_test,predicted_lin))
print('Precision:',metrics.precision_score(y_test,predicted_lin))
print('Recall:',metrics.recall_score(y_test,predicted_lin))
```

```
Accuracy: 0.956140350877193
Precision: 0.9696969696969697
Recall: 0.9552238805970149
```

In [36]:
```
print('Accuracy:',accuracy_score(y_test,predicted_rbf))
print('Precision:',metrics.precision_score(y_test,predicted_rbf))
print('Recall:',metrics.recall_score(y_test,predicted_rbf))
```

```
Accuracy: 0.956140350877193
Precision: 1.0
Recall: 0.9253731343283582
```

In [37]:
```
print('Accuracy:',accuracy_score(y_test,predicted_ply))
print('Precision:',metrics.precision_score(y_test,predicted_ply))
print('Recall:',metrics.recall_score(y_test,predicted_ply))
```

```
Accuracy: 0.9736842105263158
Precision: 0.9705882352941176
Recall: 0.9850746268656716
```

In [ ]:
```
#Q2
```

In [57]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import warnings

from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.decomposition import IncrementalPCA
from sklearn.metrics import classification_report
from sklearn.svm import SVR
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_recall_curve
from sklearn import metrics
from sklearn.metrics import mean_squared_error
```

In [58]:
```
warnings.filterwarnings('ignore')
```

In [59]:
```
dataset = pd.read_csv('Housing.csv')
```

```
df = dataset.drop('furnishingstatus', axis=1)
```

In [60]:
```
svar_list = ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning', 'prefarea']
def binary_mapping(x):
    return x.map({'yes':1, 'no':0})
df[svar_list] = df[svar_list].apply(binary_mapping)

df.head()
```

Out[60]:

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

In [61]:
```
X = df[['area','bedrooms','bathrooms','stories','mainroad','guestroom','basement','hotwaterheating','airconditio
y = dataset['price']

X.shape
```

Out[61]:
```
(545, 11)
```

In [62]:
```
y.shape
```

Out[62]:
```
(545,)
```

In [67]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.8, test_size = 0.2, random_state = 0)

sc = StandardScaler()

Xsc_train = sc.fit_transform(X_train)
Xsc_test = sc.transform(X_test)
```

In [81]:
```
svr_rbf = SVR(kernel='rbf', C=1e6, gamma=0.1)
svr_lin = SVR(kernel='linear', C=1e6)
svr_ply = SVR(kernel='poly', C=1e6, degree=2)

y_rbf = svr_rbf.fit(Xsc_train,y_train).predict(Xsc_test)
y_lin = svr_lin.fit(Xsc_train,y_train).predict(Xsc_test)
y_ply = svr_ply.fit(Xsc_train,y_train).predict(Xsc_test)

print('Loss:', mean_squared_error(y_test, y_lin), mean_squared_error(y_test, y_rbf), mean_squared_error(y_test,
```

```
Loss: 947860338641.9976 1279027590158.7593 1622321986946.383
```

In [82]:
```
model = PCA()

model.fit(Xsc_train)
```

Out[82]:
```
PCA()
```

In [83]:
```
PCA_train = IncrementalPCA(n_components = 11)
df_train = PCA_train.fit_transform(Xsc_train)
df_test = PCA_train.transform(Xsc_test)
```

In [86]:
```
y_rbf_pca = svr_rbf.fit(df_train,y_train).predict(X_test)
y_lin_pca = svr_lin.fit(df_train,y_train).predict(X_test)
y_ply_pca = svr_ply.fit(df_train,y_train).predict(X_test)
```

In [90]:
```
print('Loss:', mean_squared_error(y_test, y_lin_pca), mean_squared_error(y_test, y_rbf_pca), mean_squared_error(
```

```
Loss: 2.662630291102947e+19 3086028271056.9614 4.5372558100900655e+25
```

In [ ]:
```

```