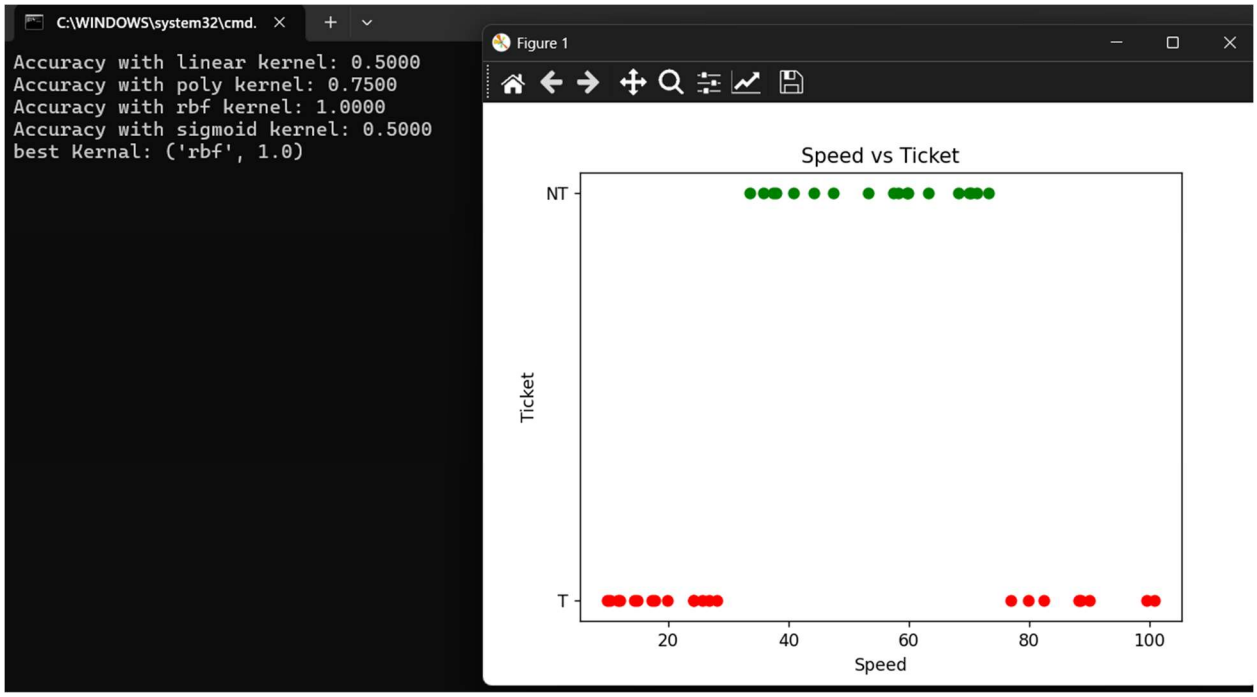
1.

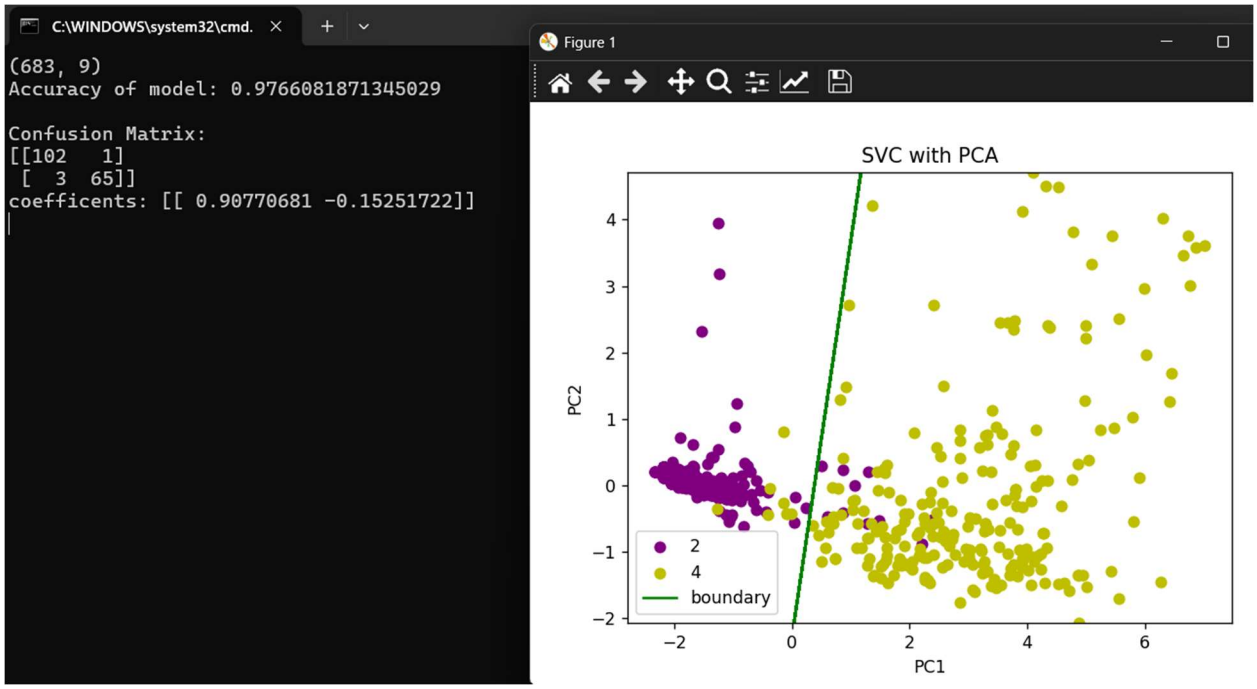
```
from sklearn.metrics import accuracy_score
import warnings

# Suppress all warnings
warnings.filterwarnings("ignore")

data = pd.read_csv('speedLimits.csv')

X =  np.array(data['Speed']).reshape(-1, 1)
y =  np.array(data['Ticket']).reshape(-1, 1)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=0)

# Define different kernels
kernels = ['linear', 'poly', 'rbf', 'sigmoid']
classifiers = {}

# Train SVM models with different kernels
for kernel in kernels:
    clf = SVC(kernel=kernel, C=1.0)
    clf.fit(X_train, y_train)
    classifiers[kernel] = clf

# Evaluate the models
accuracies = {}
for kernel, clf in classifiers.items():
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    accuracies[kernel] = accuracy
    print(f"Accuracy with {kernel} kernel: {accuracy:.4f}")

#print best
print(f'best Kernal: {max(accuracies.items(), key=lambda k: k[1])}')

#plot
for i in range(len(data)):
    if data.iloc[i, 1] == 'NT':
        plt.scatter(data.iloc[i,0],data.iloc[i,1], color = 'g')
    else:
        plt.scatter(data.iloc[i,0],data.iloc[i,1], color = 'r')

plt.title('Speed vs Ticket')
plt.xlabel('Speed')
plt.ylabel('Ticket')
plt.show()

```


C:\WINDOWS\system32\cmd.
Accuracy with linear kernel: 0.5000
Accuracy with poly kernel: 0.7500
Accuracy with rbf kernel: 1.0000
Accuracy with sigmoid kernel: 0.5000
best Kernal: ('rbf', 1.0)
Figure 1
Speed vs Ticket
NT
T
Ticket
20
40
60
80
100
Speed

2.

```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import warnings
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import accuracy_score, confusion_matrix

# Suppress all warnings
warnings.filterwarnings("ignore")

data = pd.read_csv('breast-cancer-wisconsin-data.csv', header=None)
#clean data
data.replace('?', None, inplace = True)
data = data.dropna()

X =  np.array(data.iloc[:, 1:10])
print(X.shape)
y =  np.array(data.iloc[:,10]).reshape(-1, 1)

#standardize
scaler = StandardScaler()
X = scaler.fit_transform(X)

#PCA
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(X)
principalDf = pd.DataFrame(data = principalComponents, columns = ['principalcomponent 1', 'principalcomponent 2'])
principalDf['Classes'] = y

X_train, X_test, y_train, y_test = train_test_split(principalComponents, y, test_size=0.25, random_state=42)

clf = SVC(kernel='linear')  # Soft-margin with default C=1.0
clf.fit(X_train, y_train)
y_pred =  clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy of model: {accuracy}')
print(f'\nConfusion Matrix: \n{confusion_matrix(y_test, y_pred)}')

print(f'coefficents: {clf.coef_}')
yModel = -1*((clf.coef_[0,0]*principalDf['principalcomponent 1'] + clf.intercept_) /clf.coef_[0, 1])

```

```
#plot

class1 =  principalDf[principalDf['Classes'] == 2]
class2 =  principalDf[principalDf['Classes'] == 4]

plt.scatter(class1['principalcomponent 1'], class1['principalcomponent 2'], color = 'purple', label ='2')
plt.scatter(class2['principalcomponent 1'], class2['principalcomponent 2'], color = 'y', label ='4')
plt.plot(principalDf['principalcomponent 1'], yModel, c = 'g', label='boundary')
plt.legend(loc = 'lower left')
plt.ylim(min(principalDf['principalcomponent 2']), max(principalDf['principalcomponent 2']))
plt.title('SVC with PCA')
plt.xlabel('PC1')
plt.ylabel('PC2')

plt.show()

```


C:\WINDOWS\system32\cmd.
(683, 9)
Accuracy of model: 0.9766081871345029
Confusion Matrix:
[[102 1]
[3 65]]
coefficents: [[0.90770681 -0.15251722]]
Figure 1
SVC with PCA
PC2
PC1
2
4
boundary