10/10

Allay Sukant Sahoo
B Tech CSE (A)
36.

In [ ]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets pr
# You can also write temporary files to /kaggle/temp/, but they won't be saved outs
```

```
/kaggle/input/data1asasas/1.01. Simple linear regression.csv
```

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
```

In [ ]:
```
data = pd.read_csv('/kaggle/input/data1asasas/1.01. Simple linear regression.csv')
```

In [ ]:
```
data
```

Out[ ]:

	SAT	GPA
0	1714	2.40
1	1664	2.52
2	1760	2.54
3	1685	2.74
4	1693	2.83
...	...	...
79	1936	3.71
80	1810	3.71
81	1987	3.73
82	1962	3.76
83	2050	3.81

84 rows × 2 columns

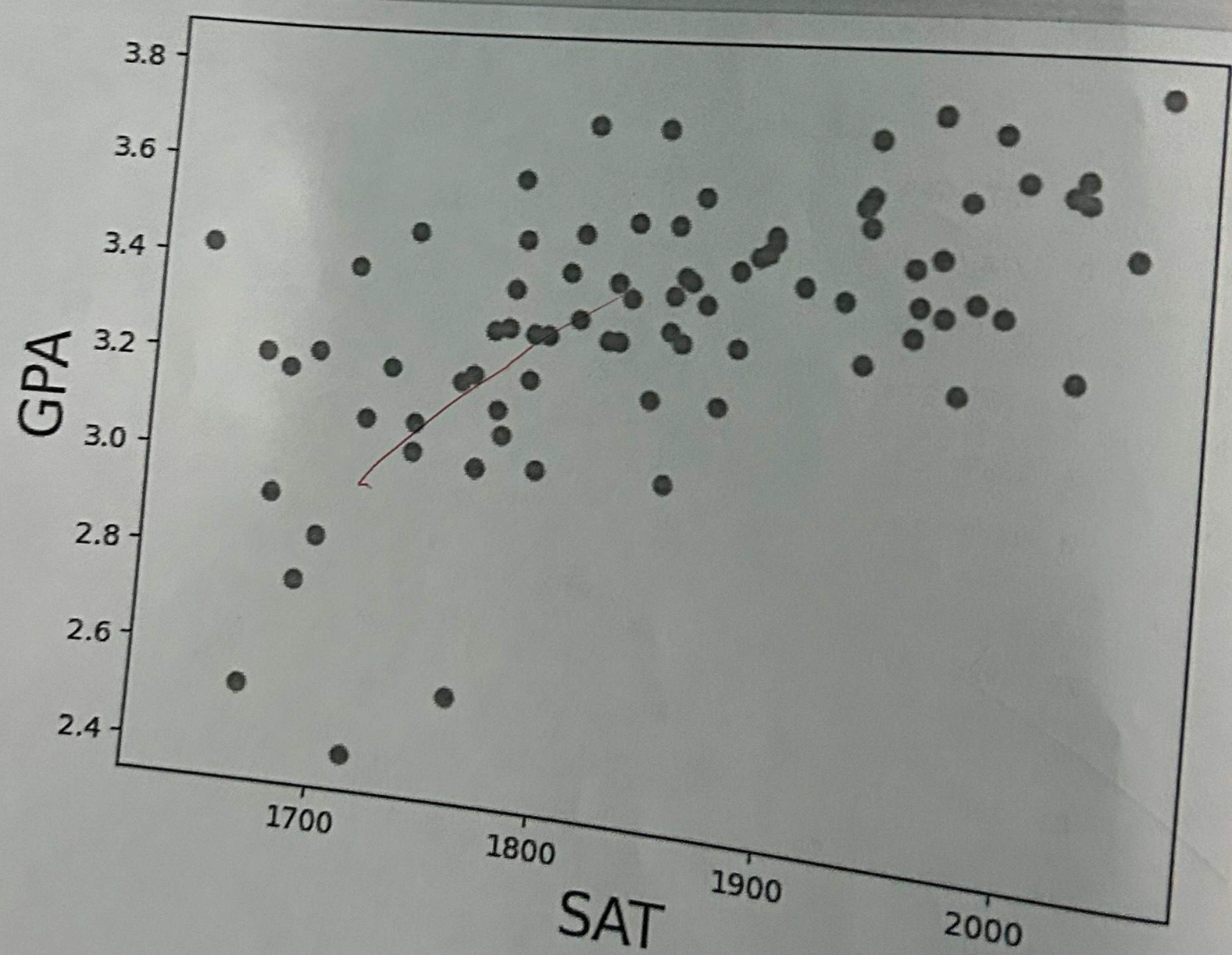
In [ ]:
```
data.describe()
```

Out[ ]:

	SAT	GPA
count	84.000000	84.000000
mean	1845.273810	3.330238
std	104.530661	0.271617
min	1634.000000	2.400000
25%	1772.000000	3.190000
50%	1846.000000	3.380000
75%	1934.000000	3.502500
max	2050.000000	3.810000

In [ ]:
```
y = data['GPA']
x1 = data['SAT']
```

In [ ]:
```
plt.scatter(x1,y)
plt.xlabel('SAT', fontsize = 20)
plt.ylabel('GPA', fontsize = 20)
plt.show()
```



In [ ]:
```
x = sm.add_constant(x1)
```

In [ ]:
```
x = sm.add_constant(x1)
results = sm.OLS(y,x).fit()
results.summary()
```

Out[ ]:

OLS Regression Results

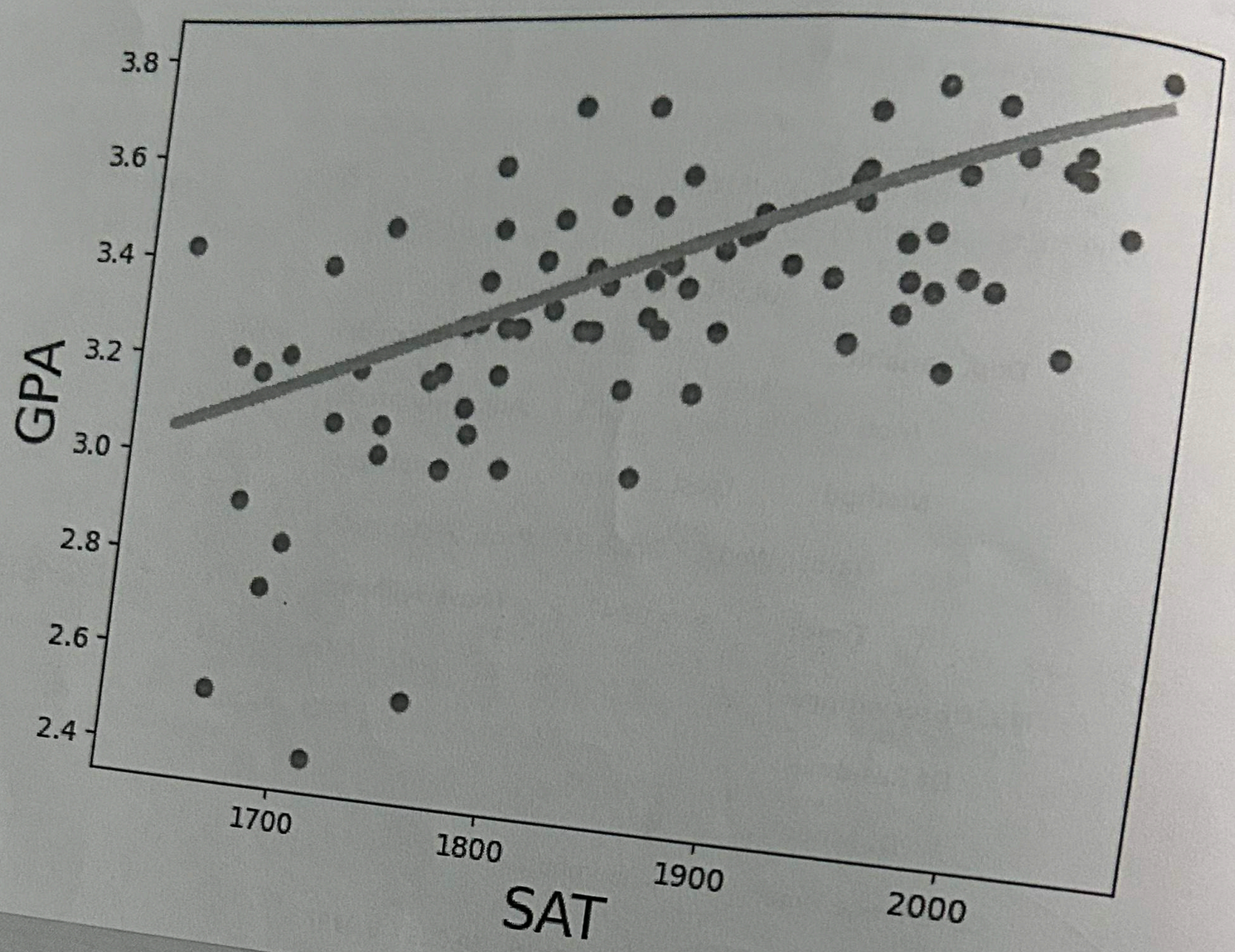
Dep. Variable:	GPA	**R-squared:**	0.406
Model:	OLS	**Adj. R-squared:**	0.399
Method:	Least Squares	**F-statistic:**	56.05
Date:	Wed, 25 Oct 2023	**Prob (F-statistic):**	7.20e-11
Time:	09:47:25	**Log-Likelihood:**	12.672
No. Observations:	84	**AIC:**	-21.34
Df Residuals:	82	**BIC:**	-16.48
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	0.2750	0.409	0.673	0.503	-0.538	1.088
SAT	0.0017	0.000	7.487	0.000	0.001	0.002

Omnibus:	12.839	**Durbin-Watson:**	0.950
Prob(Omnibus):	0.002	**Jarque-Bera (JB):**	16.155
Skew:	-0.722	**Prob(JB):**	0.000310
Kurtosis:	4.590	**Cond. No.**	3.29e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.29e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

In [ ]:
```
plt.scatter(x1,y)
yhat = 0.0017*x1 + 0.275
fig = plt.plot(x1,yhat, lw=4, c='orange', label = 'regressionline')
plt.xlabel('SAT', fontsize = 20)
plt.ylabel('GPA', fontsize = 20)
plt.show()
```



In [ ]:

In [1]:
```
import pandas as pd
data = pd.read_csv("/kaggle/input/apples-and-oranges/apples_and_oranges.csv")
```

In [3]:
```
from sklearn.model_selection import train_test_split
training_set, test_set = train_test_split(data, test_size = 0.2, random_state =
```

In [4]:
```
X_train = training_set.iloc[:,0:2].values
Y_train = training_set.iloc[:,2].values
X_test = test_set.iloc[:,0:2].values
Y_test = test_set.iloc[:,2].values
```

In [5]:
```
from sklearn.svm import SVC
classifier = SVC(kernel='rbf', random_state = 1)
classifier.fit(X_train,Y_train)
```

Out[5]:
```
▾ SVC
SVC(random_state=1)
```

In [6]:
```
#Predicting the classes for test set
Y_pred = classifier.predict(X_test)
#Attaching the predictions to test set for comparing
test_set["Predictions"] = Y_pred
#Comparing the actual classes and predictions
```

In [7]:
```
test_set["Predictions"]
```

Out[7]:
```
2     apple
31    apple
3     apple
21    apple
27    apple
29    apple
22    apple
39    apple
Name: Predictions, dtype: object
```

In [8]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test,Y_pred)
accuracy = float(cm.diagonal().sum())/len(Y_test)
print("\nAccuracy Of SVM For The Given Dataset : ", accuracy)
```

```
Accuracy Of SVM For The Given Dataset :  0.375
```

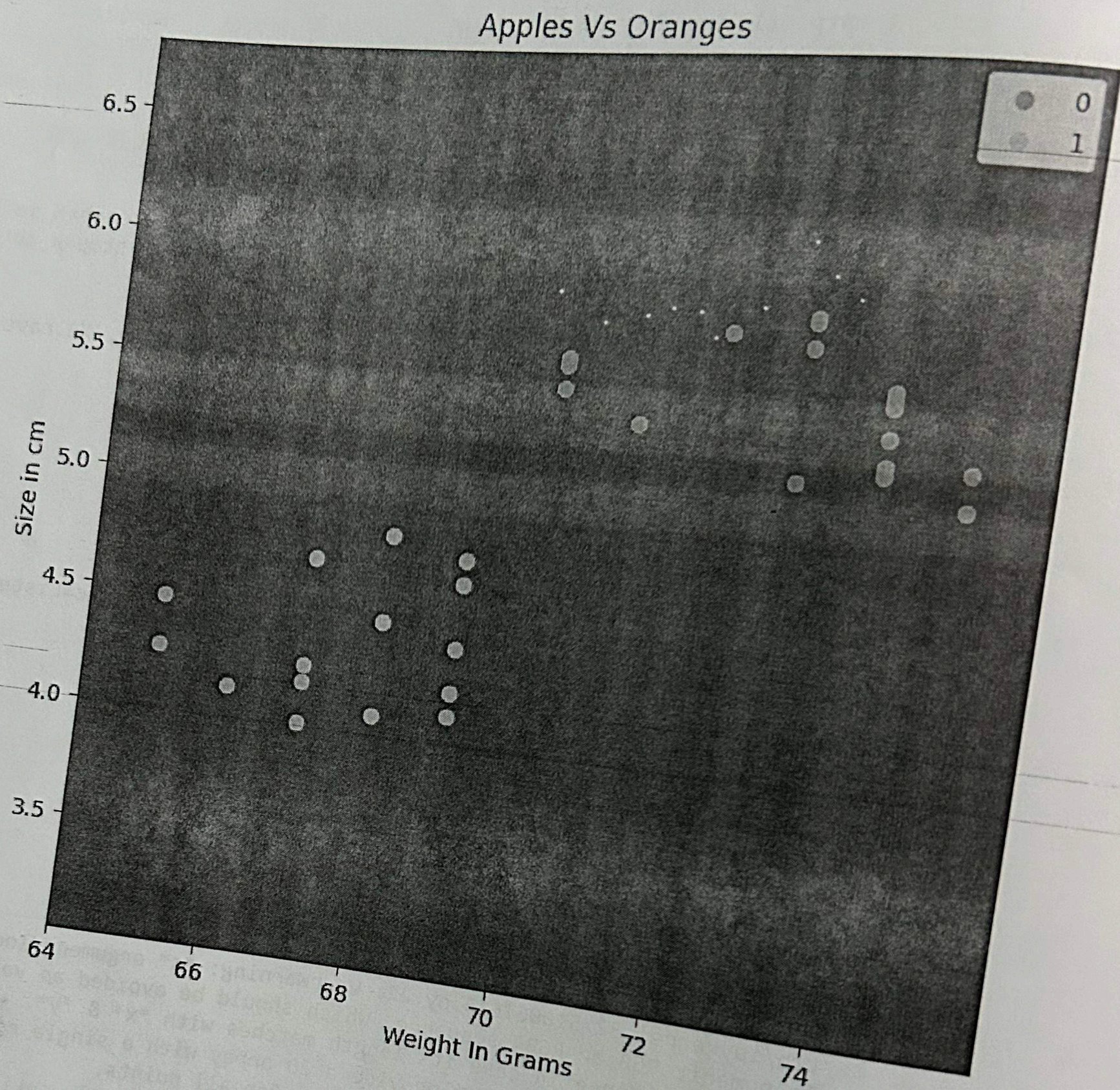
In [9]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
Y_train = le.fit_transform(Y_train)
```

In [10]:
```
from sklearn.svm import SVC
classifier = SVC(kernel='rbf', random_state = 1)
classifier.fit(X_train,Y_train)
```

Out[10]:
```
SVC
SVC(random_state=1)
```

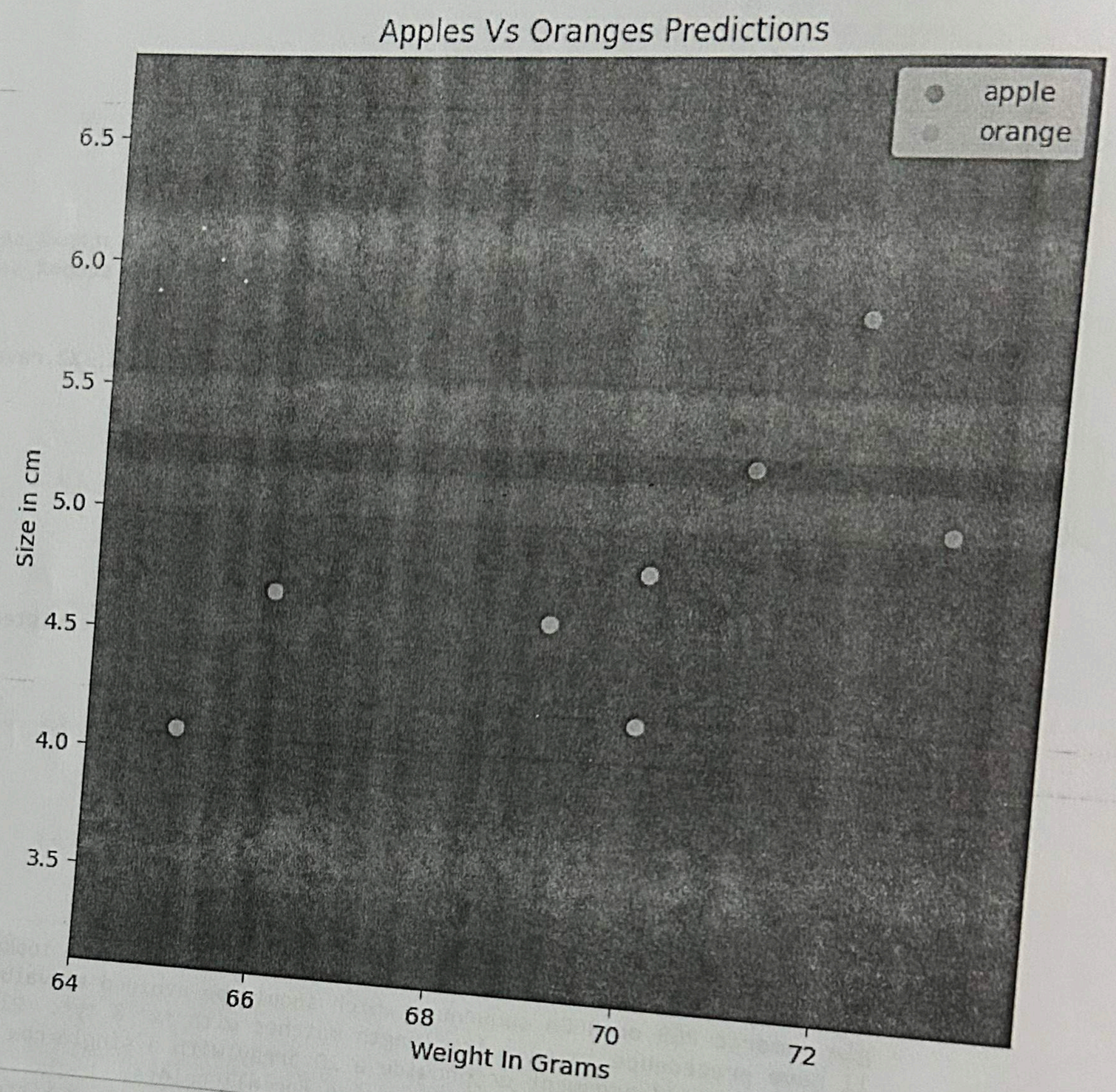
In [11]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

plt.figure(figsize=(7, 7))

X_set, y_set = X_train, Y_train

X1, X2 = np.meshgrid(np.arange(start=X_set[:, 0].min() - 1, stop=X_set[:, 0].ma
                     np.arange(start=X_set[:, 1].min() - 1, stop=X_set[:, 1].ma

# Create a contour plot for the decision boundary
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).r
             cmap=ListedColormap(('black', 'white')))

# Set the plot limits
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())

# Create a scatter plot of the data points
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c=ListedColormap((

# Set the title, labels, and legend
plt.title('Apples Vs Oranges')
plt.xlabel('Weight In Grams')
plt.ylabel('Size in cm')
plt.legend()

# Show the plot
plt.show()
```

```
/tmp/ipykernel_32/510067616.py:22: UserWarning: *c* argument looks like a sin
gle numeric RGB or RGBA sequence, which should be avoided as value-mapping wi
ll have precedence in case its length matches with *x* & *y*.  Please use the
*color* keyword-argument or provide a 2D array with a single row if you inten
d to specify the same RGB or RGBA value for all points.
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c=ListedColormap
(('red', 'orange'))(i), label=j)
```


Apples Vs Oranges
0
1
6.5
6.0
5.5
5.0
4.5
4.0
3.5
Size in cm
64
66
68
70
72
74
Weight In Grams

In [13]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

plt.figure(figsize=(7, 7))

X_set, y_set = X_test, Y_test

X1, X2 = np.meshgrid(np.arange(start=X_set[:, 0].min() - 1, stop=X_set[:, 0].ma
                     np.arange(start=X_set[:, 1].min() - 1, stop=X_set[:, 1].ma

# Create a contour plot for the decision boundary
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).r
             cmap=ListedColormap(('black', 'white')))

# Set the plot limits
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())

# Create a scatter plot of the data points
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c=ListedColormap((

# Set the title, labels, and legend
plt.title('Apples Vs Oranges Predictions')
plt.xlabel('Weight In Grams')
plt.ylabel('Size in cm')
plt.legend()

# Show the final plot
plt.show()
```

```
/tmp/ipykernel_32/540414344.py:22: UserWarning: *c* argument looks like a sin
gle numeric RGB or RGBA sequence, which should be avoided as value-mapping wi
ll have precedence in case its length matches with *x* & *y*.  Please use the
*color* keyword-argument or provide a 2D array with a single row if you inten
d to specify the same RGB or RGBA value for all points.
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c=ListedColormap
(('red', 'orange'))(i), label=j)
```


Apples Vs Oranges Predictions
apple
orange
Size in cm
6.5
6.0
5.5
5.0
4.5
4.0
3.5
64
66
68
70
72
Weight In Grams

In [ ]:

3/11/23