```
1 import pandas as pd
2 import numpy as np
3
```

```
1 cell_df = pd.read_csv("/content/cell_samples.csv")
2 cell_df.head()
3
```

	ID	Clump	UnifSize	UnifShape	MargAdh	SingEpiSize	BareNuc	BlandChrom	Nc
0	1000025	5	1	1	1	2	1	3	
1	1002945	5	4	4	5	7	10	3	
2	1015425	3	1	1	1	2	2	3	
3	1016277	6	8	8	1	3	4	3	
4	1017023	4	1	1	3	2	1	3	

```
1 cell_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 699 entries, 0 to 698
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   ID           699 non-null    int64
 1   Clump        699 non-null    int64
 2   UnifSize     699 non-null    int64
 3   UnifShape    699 non-null    int64
 4   MargAdh      699 non-null    int64
 5   SingEpiSize  699 non-null    int64
 6   BareNuc      699 non-null    object
 7   BlandChrom   699 non-null    int64
 8   NormNucl     699 non-null    int64
 9   Mit          699 non-null    int64
 10  Class        699 non-null    int64
dtypes: int64(10), object(1)
memory usage: 60.2+ KB
```

```
1 cell_df['BareNuc'].unique()
```

```
array(['1', '10', '2', '4', '3', '9', '7', '?', '5', '8', '6'],
      dtype=object)
```

```
1 type(cell_df['BareNuc'][0])
```

```
str
```

```
1 # It looks like the BareNuc column includes some values that are not numerical. We can
```

```
cell_df = cell_df[pd.to_numeric(cell_df['BareNuc'], errors='coerce').notnull()] #object

```

```
cell_df['BareNuc'] = cell_df['BareNuc'].astype('int')# numeric to integer
cell_df.dtypes
```

```
ID             int64
Clump          int64
UnifSize       int64
UnifShape      int64
MargAdh        int64
SingEpiSize    int64
BareNuc        int64
BlandChrom     int64
NormNucl       int64
Mit            int64
Class          int64
dtype: object
```

```
feature_df = cell_df[['Clump', 'UnifSize', 'UnifShape', 'MargAdh', 'SingEpiSize', 'Bare
feature_df
```

	Clump	UnifSize	UnifShape	MargAdh	SingEpiSize	BareNuc	BlandChrom	NormNucl
0	5	1	1	1	2	1	3	1
1	5	4	4	5	7	10	3	2
2	3	1	1	1	2	2	3	1
3	6	8	8	1	3	4	3	7
4	4	1	1	3	2	1	3	1
...	...	...	...	...	...	...	...	...
694	3	1	1	1	3	2	1	1
695	2	1	1	1	2	1	1	1
696	5	10	10	3	7	3	8	10
697	4	8	6	4	3	4	10	6
698	4	8	8	5	4	5	10	4

683 rows × 9 columns

```
X = np.asarray(feature_df) #independet variable independent variable array got created
X[0:5] #show me elements from zeroth row to 5th row
```

```
array([[ 5,  1,  1,  1,  2,  1,  3,  1,  1],
       [ 5,  4,  4,  5,  7, 10,  3,  2,  1],
       [ 3,  1,  1,  1,  2,  2,  3,  1,  1],
       [ 6,  8,  8,  1,  3,  4,  3,  7,  1],
       [ 4,  1,  1,  3,  2,  1,  3,  1,  1]])
```

```
cell_df['Class'] = cell_df['Class'].astype('int')
```

```
y = np.asarray(cell_df['Class']) #dependent variable
y [0:5]
```

```
array([2, 2, 2, 2, 2])
```

```
cell_df['Class'].unique()
```

```
array([2, 4])
```

```
#  split the data to train and test
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=5
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (546, 9) (546,)
Test set: (137, 9) (137,)
```

```
from sklearn import svm
clf = svm.SVC(kernel='poly')
clf.fit(X_train, y_train) # question and answers
```

```
SVC(kernel='poly')
```

```
yhat = clf.predict(X_test) #question
yhat [0:5]
```

```
array([4, 4, 2, 2, 4])
```

```
y_test[:5]
```

```
array([4, 4, 2, 2, 4])
```

```
# evaluation
from sklearn.metrics import accuracy_score, f1_score
accuracy_score(y_test, yhat)
```

```
0.9708029197080292
```

```
# using 'rbf' kernel

clf2 = svm.SVC(kernel='rbf')
clf2.fit(X_train, y_train)
yhat2 = clf2.predict(X_test)
print("Avg F1-score: %.4f" % f1_score(y_test, yhat2, average='weighted'))
#print("Jaccard score: %.4f" % jaccard_similarity_score(y_test, yhat2))
```

```
Avg F1-score: 0.9854
```

```
clf2.predict([[6,4,2,3,1,2,5,2,1]])                    # 2= benign , 4= malignant
```

```
array([2])
```

the user would just give the input data. we need to preprocess the data in the right format. so below code is written for the same to handle new given data

```
input_data = [[9, 10, 10, 1, 10, 8, 3, 3, 1]]                # im using the svm model

prediction = clf2.predict(input_data)
print(prediction)

if (prediction[0] == 2):
  print('Its a benign cancer')
else:
  print('Its a malignant cancer, Consult doctor immediately')
```

```
[4]
Its a malignant cancer, Consult doctor immediately
```

```
input_data = (9, 10, 10, 1, 10, 8, 3, 3, 1)
input_data_arr= np.asarray(input_data)
inputdata_reshaped= input_data_arr.reshape(1, -1)              # coz it sontains sampl

prediction = clf2.predict(inputdata_reshaped)             # here instead of clf2, im g
print(prediction)

if (prediction[0] == 2):
  print('Its a benign cancer')
else:
  print('Its a malignant cancer, Consult doctor immediately')
```

```
[4]
Its a malignant cancer, Consult doctor immediately
```

We are done with model. Now lets move ahead with deployment.

Saving the trained model

```
import pickle                                          # pickle libraray is a need while d
```

```
filename= 'trained_model.sav'
pickle.dump(clf2, open(filename, 'wb'))

# i created a variable name 'filename' where i want my trained model to be in.
# then i dump the trained model(loaded in clf2 or clf) to the filename created through
```

Loading the saved model

```
loaded_model= pickle.load(open('trained_model.sav', 'rb'))

# here im loading the model to a variable. Im'm giving the trained model name, u can al
# this time i want to read the file so 'rb
```

```
input_data = (6, 4, 2, 3, 1, 2, 5, 2, 1)
input_data_arr= np.asarray(input_data)
inputdata_reshaped= input_data_arr.reshape(1, -1)              # coz it sontains sampl

prediction = loaded_model.predict(inputdata_reshaped)             # here instead of cl
print(prediction)

if (prediction[0] == 2):
  print('Its a benign cancer')
else:
  print('Its a malignant cancer, Consult doctor immediately')


```

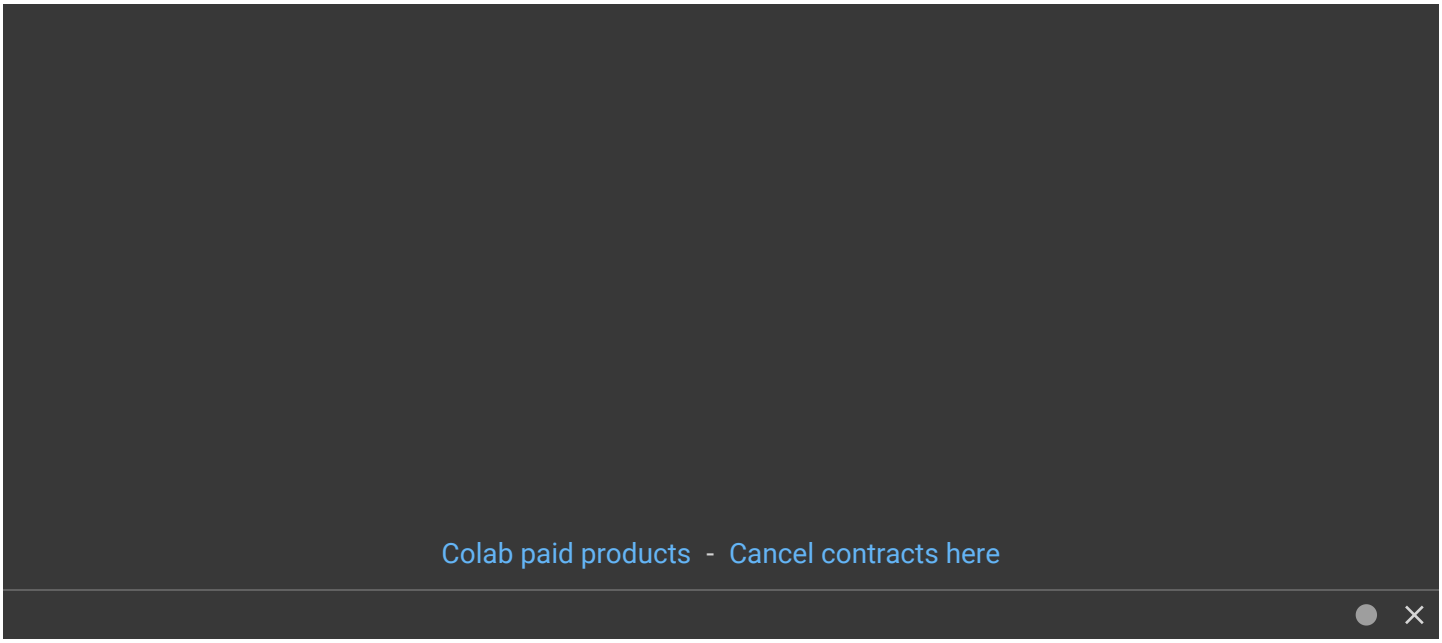
```
[2]
Its a benign cancer
```

```
# Whatever the input data we have given we want the data to come from the user when we
# so here we need to work on the parameters that can will avaiable on user interface fo
# so that the web app is able to predict the kind of cancer the patient has ..

# rest of teh code will be writte on spyder tool. before that lets download the trained
```

```
1
```

```
1
```


Colab paid products - Cancel contracts here