In [1]:

```python
# !pip install pyforest
from pyforest import *
```

In [2]:

```python
lazy_imports()
```

Out[2]:

```
['import re',
 'import fastai',
 'from sklearn.manifold import TSNE',
 'from sklearn import svm',
 'import matplotlib as mpl',
 'import lightgbm as lgb',
 'from scipy import stats',
 'from xlrd import open_workbook',
 'from sklearn.preprocessing import StandardScaler',
 'import spacy',
 'from sklearn.preprocessing import MinMaxScaler',
 'from pathlib import Path',
 'import numpy as np',
 'from statsmodels.tsa.arima_model import ARIMA',
 'import plotly.express as px',
 'import glob',
 'import pandas as pd',
 'from dask import dataframe as dd',
 'from sklearn.linear_model import Ridge',
 'import tqdm',
 'import statistics',
 'import pydot',
 'from sklearn.linear_model import ElasticNet',
 'from sklearn.linear_model import ElasticNetCV',
 'import textblob',
 'from sklearn.ensemble import RandomForestRegressor',
 'from sklearn.preprocessing import RobustScaler',
 'from PIL import Image',
 'from sklearn.model_selection import RandomizedSearchCV',
 'from sklearn.model_selection import GridSearchCV',
 'from sklearn.cluster import KMeans',
 'import tensorflow as tf',
 'import plotly.graph_objs as go',
 'import plotly as py',
 'from sklearn.linear_model import Lasso',
 'import statsmodels.api as sm',
 'from sklearn.ensemble import GradientBoostingClassifier',
 'import fbprophet',
 'import seaborn as sns',
 'from sklearn.model_selection import StratifiedKFold',
 'from sklearn.preprocessing import PolynomialFeatures',
 'from openpyxl import load_workbook',
 'import sys',
 'from sklearn.model_selection import cross_val_score',
 'from pyspark import SparkContext',
 'import matplotlib.pyplot as plt',
 'from sklearn.linear_model import LinearRegression',
 'import awswrangler as wr',
 'import datetime as dt',
 'import imutils',
 'from sklearn.linear_model import RidgeCV',
 'import nltk',
 'import os',
 'import cv2',
 'from sklearn.decomposition import PCA',
 'from sklearn.feature_extraction.text import CountVectorizer',
 'from sklearn.linear_model import LogisticRegression',
 'import skimage',
 'from fbprophet import Prophet',
 'import keras',
 'import torch',
 'import sklearn',
 'from sklearn.preprocessing import OneHotEncoder',
 'from sklearn.model_selection import train_test_split',
 'from sklearn.ensemble import GradientBoostingRegressor',
 'import bokeh',
 'from sklearn.impute import SimpleImputer',
 'from sklearn import metrics',
 'from sklearn.linear_model import LassoCV',
 'from sklearn.ensemble import RandomForestClassifier',
 'import gensim',
 'import altair as alt',
 'from sklearn.feature_extraction.text import TfidfVectorizer',
 'import pickle',
 'from sklearn.model_selection import KFold',
 'import dash',
 'from sklearn.preprocessing import LabelEncoder',
 'from scipy import signal as sg',
 'import xgboost as xgb']
```

In [3]:

```python
data = pd.read_csv('breastCancer.csv')
data.head()
```

Out[3]:

	id	clump_thickness	size_uniformity	shape_uniformity	marginal_adhesion	epithelial_size
0	1000025	5	1	1	1	2
1	1002945	5	4	4	5	7
2	1015425	3	1	1	1	2
3	1016277	6	8	8	1	3
4	1017023	4	1	1	3	2

In [4]:

```python
data.drop(['id'], axis=1, inplace=True)
```

In [5]:

```python
data.shape
```

Out[5]:

```
(699, 10)
```

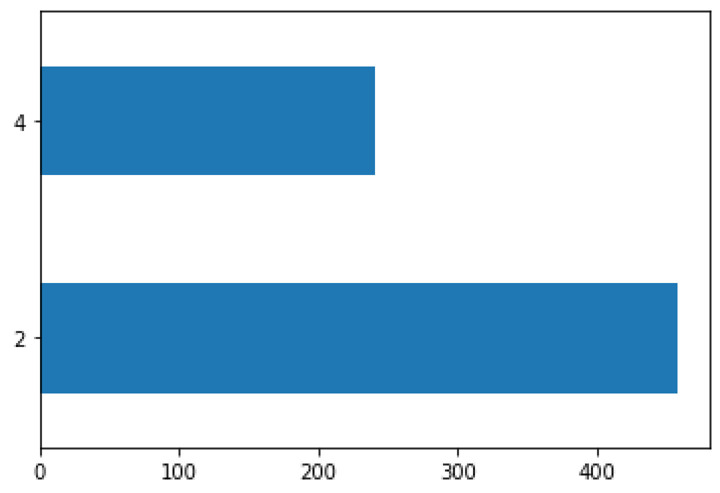
Data Preprocessing

In [6]:

```python
data['class'].value_counts().plot(kind='barh')
```

Out[6]:

```
<AxesSubplot:>
```



In [7]:
```
data.dtypes
```

Out[7]:
```
clump_thickness        int64
size_uniformity        int64
shape_uniformity       int64
marginal_adhesion      int64
epithelial_size        int64
bare_nucleoli         object
bland_chromatin        int64
normal_nucleoli        int64
mitoses                int64
class                  int64
dtype: object
```

In [8]:
```
data['bare_nucleoli'].value_counts()
```

Out[8]:
```
1     402
10    132
2      30
5      30
3      28
8      21
4      19
?      16
9       9
7       8
6       4
Name: bare_nucleoli, dtype: int64
```

In [9]:
```
data.loc[data['bare_nucleoli']=='?']
```

Out[9]:

	clump_thickness	size_uniformity	shape_uniformity	marginal_adhesion	epithelial_size	bare_nucl
23	8	4	5	1	2	
40	6	6	6	9	6	
139	1	1	1	1	1	
145	1	1	3	1	2	
158	1	1	2	1	3	
164	5	1	1	1	2	
235	3	1	4	1	2	

	clump_thickness	size_uniformity	shape_uniformity	marginal_adhesion	epithelial_size	bare_nucl
249	3	1	1	1	2	
275	3	1	3	1	2	
292	8	8	8	1	2	
294	1	1	1	1	2	
297	5	4	3	1	2	
315	4	6	5	6	7	
321	3	1	1	1	2	
411	1	1	1	1	1	
617	1	1	1	1	1	

In [10]:
```
data['bare_nucleoli'].replace('?', np.nan, inplace=True)
```

In [11]:
```
data.fillna(data.median(), inplace=True)
```

In [12]:
```
data['bare_nucleoli'] = data['bare_nucleoli'].astype('int64')
```

In [13]:
```
data.dtypes
```

Out[13]:
```
clump_thickness      int64
size_uniformity      int64
shape_uniformity     int64
marginal_adhesion    int64
epithelial_size      int64
bare_nucleoli        int64
bland_chromatin      int64
normal_nucleoli      int64
mitoses              int64
class                int64
dtype: object
```

EDA

In [14]:
```
data.describe().T
```

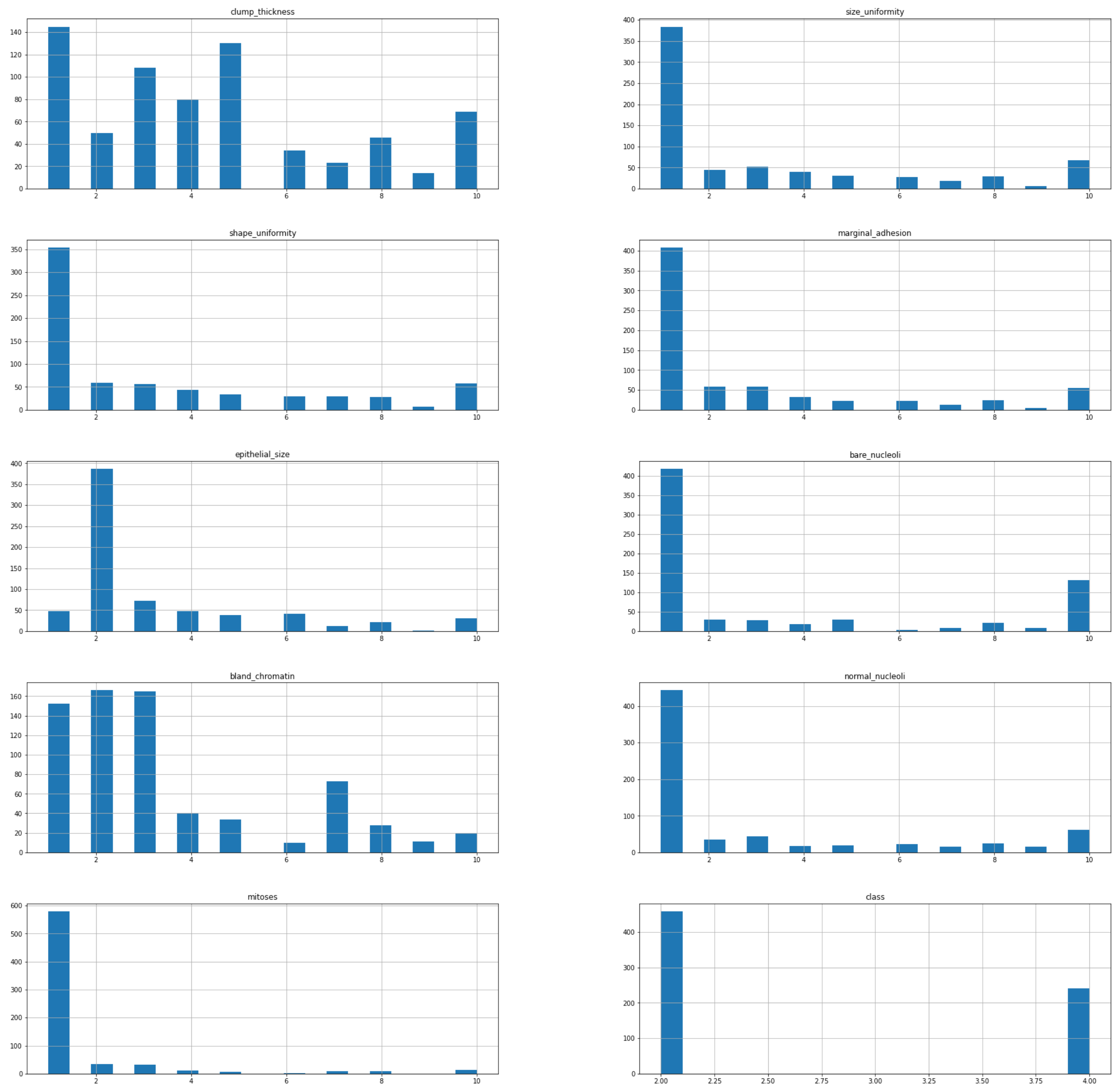
Out[14]:

	count	mean	std	min	25%	50%	75%	max
clump_thickness	699.0	4.417740	2.815741	1.0	2.0	4.0	6.0	10.0
size_uniformity	699.0	3.134478	3.051459	1.0	1.0	1.0	5.0	10.0
shape_uniformity	699.0	3.207439	2.971913	1.0	1.0	1.0	5.0	10.0
marginal_adhesion	699.0	2.806867	2.855379	1.0	1.0	1.0	4.0	10.0
epithelial_size	699.0	3.216023	2.214300	1.0	2.0	2.0	4.0	10.0
bare_nucleoli	699.0	3.486409	3.621929	1.0	1.0	1.0	5.0	10.0

	count	mean	std	min	25%	50%	75%	max
bland_chromatin	699.0	3.437768	2.438364	1.0	2.0	3.0	5.0	10.0
normal_nucleoli	699.0	2.866953	3.053634	1.0	1.0	1.0	4.0	10.0
mitoses	699.0	1.589413	1.715078	1.0	1.0	1.0	1.0	10.0
class	699.0	2.689557	0.951273	2.0	2.0	2.0	4.0	4.0

In [15]:

```
data.hist(bins=20, figsize=(30,30), layout=(5,2));
```

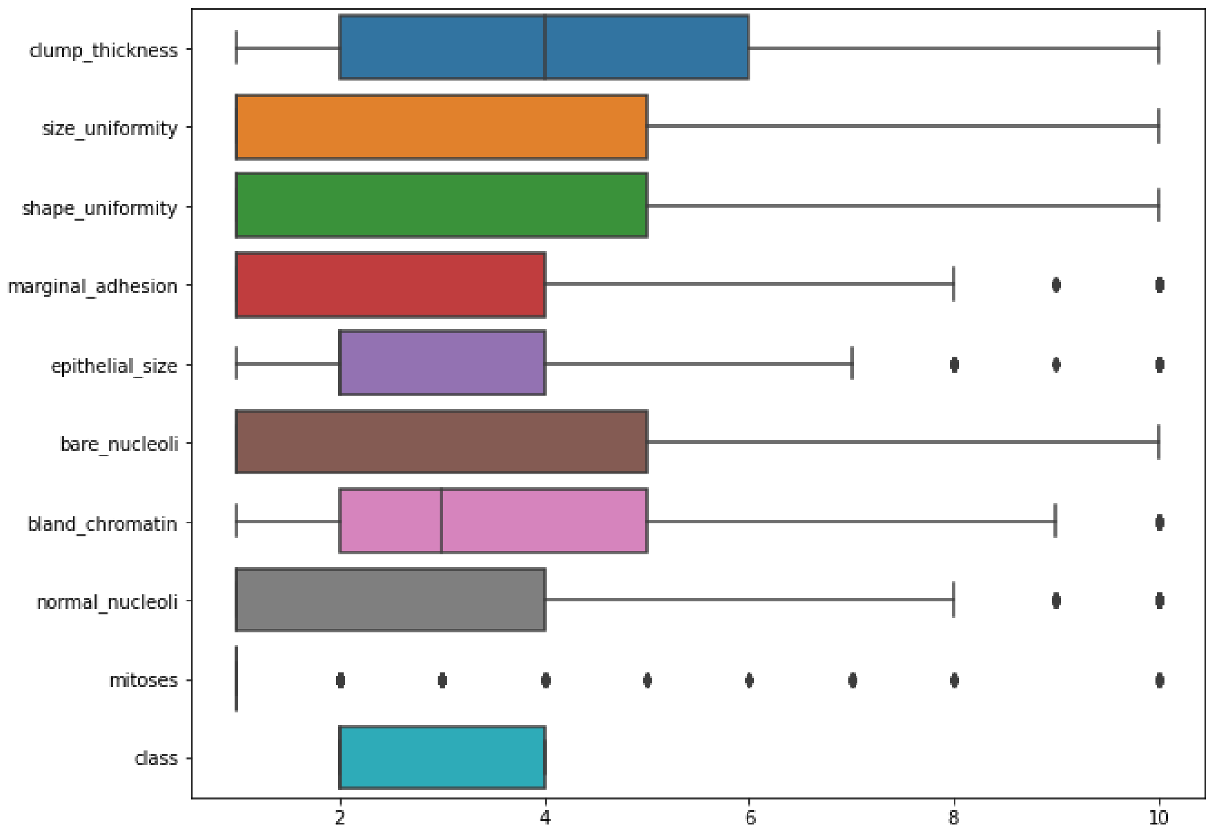


In [16]:

```
plt.figure(figsize=(10,8))
sns.boxplot(data, orient='h')
```

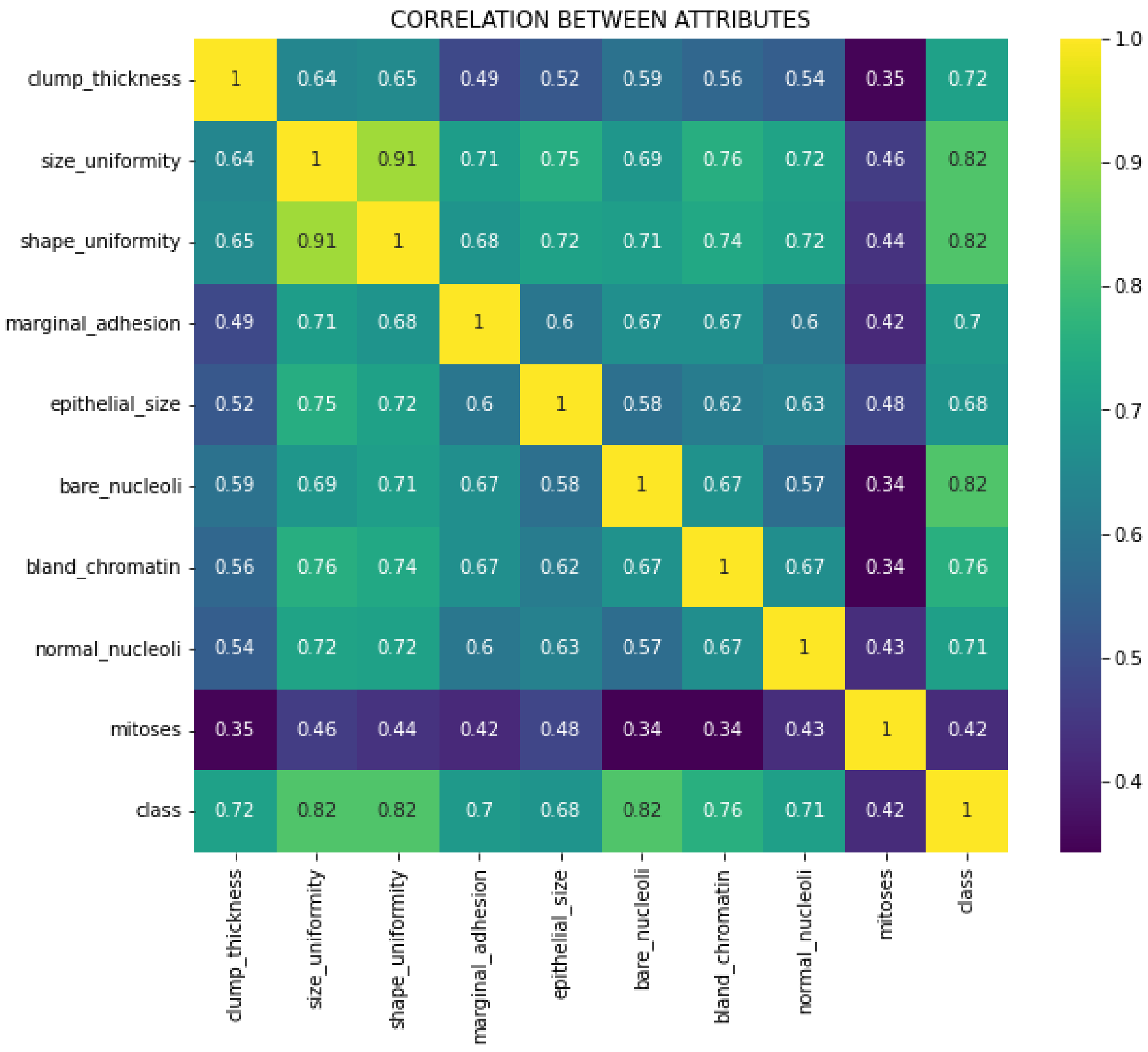
Out[16]:

```
<AxesSubplot:>
```



In [17]:

```
plt.figure(figsize=(10,8))
sns.heatmap(data.corr(), annot=True, cmap='viridis', vmax=1, square=True)
plt.title('CORRELATION BETWEEN ATTRIBUTES')
plt.show()
```



	clump_thickness	size_uniformity	shape_uniformity	marginal_adhesion	epithelial_size	bare_nucleoli	bland_chromatin	normal_nucleoli	mitoses	class
clump_thickness	1	0.64	0.65	0.49	0.52	0.59	0.56	0.54	0.35	0.72
size_uniformity	0.64	1	0.91	0.71	0.75	0.69	0.76	0.72	0.46	0.82
shape_uniformity	0.65	0.91	1	0.68	0.72	0.71	0.74	0.72	0.44	0.82
marginal_adhesion	0.49	0.71	0.68	1	0.6	0.67	0.67	0.6	0.42	0.7
epithelial_size	0.52	0.75	0.72	0.6	1	0.58	0.62	0.63	0.48	0.68
bare_nucleoli	0.59	0.69	0.71	0.67	0.58	1	0.67	0.57	0.34	0.82
bland_chromatin	0.56	0.76	0.74	0.67	0.62	0.67	1	0.67	0.34	0.76
normal_nucleoli	0.54	0.72	0.72	0.6	0.63	0.57	0.67	1	0.43	0.71
mitoses	0.35	0.46	0.44	0.42	0.48	0.34	0.34	0.43	1	0.42
class	0.72	0.82	0.82	0.7	0.68	0.82	0.76	0.71	0.42	1

Building Model

In [18]:
```
data.head()
```

Out[18]:

	clump_thickness	size_uniformity	shape_uniformity	marginal_adhesion	epithelial_size	bare_nucleo
0	5	1	1	1	2	
1	5	4	4	5	7	1
2	3	1	1	1	2	
3	6	8	8	1	3	
4	4	1	1	3	2	

In [19]:
```
X = data.drop('class', axis=1)
y = data['class']
```

In [20]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_sta
```

In [21]:
```
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```
Out[21]:
```
((524, 9), (524,), (175, 9), (175,))
```

KNN Classifier

In [22]:
```
from sklearn.neighbors import KNeighborsClassifier
from scipy.stats import zscore
```

In [23]:
```
knn = KNeighborsClassifier(n_neighbors=5, weights='distance')
```

In [24]:
```
knn.fit(X_train, y_train)
```
Out[24]:
```
▾ KNeighborsClassifier
KNeighborsClassifier(weights='distance')
```

In [25]:
```
knn_preds = knn.predict(X_test)
```

In [26]:
```
print('KNN Model accuracy: ', knn.score(X_test, y_test))
```
```
KNN Model accuracy:  0.9771428571428571
```

SVC Model

In [27]:
```
svc = svm.SVC(C=3)
```

In [28]:
```
svc.fit(X_train, y_train)
```
Out[28]:
```
▾ SVC
SVC(C=3)
```

In [29]:
```
svc_preds = svc.predict(X_test)
```

In [30]:
```
print('SVC Model accuracy: ', svc.score(X_test, y_test))
```
```
SVC Model accuracy:  0.9542857142857143
```

Comparing both Models

In [31]:
```
models = pd.concat([pd.DataFrame(knn_preds), pd.DataFrame(svc_preds)], axis=1)
```

In [32]:
```
models.columns = ['knn_preds', 'svc_preds']
```

In [33]:
```
models
```

Out[33]:

	knn_preds	svc_preds
0	2	2
1	2	2
2	2	2
3	4	4
4	2	2
...	...	...
170	2	2
171	2	2
172	4	4
173	4	4
174	2	2

175 rows × 2 columns

In [34]:
```
from sklearn.metrics import classification_report
```

In [35]:
```
print('KNN Classification Report')
print('......'*10)
print(classification_report(y_test, knn_preds))
```

```
KNN Classification Report
............................................................
              precision    recall  f1-score   support

           2       0.98      0.98      0.98       118
           4       0.96      0.96      0.96        57

    accuracy                           0.98       175
   macro avg       0.97      0.97      0.97       175
weighted avg       0.98      0.98      0.98       175
```

In [36]:
```
print('SVC Classification Report')
print('......'*10)
print(classification_report(y_test, svc_preds))
```

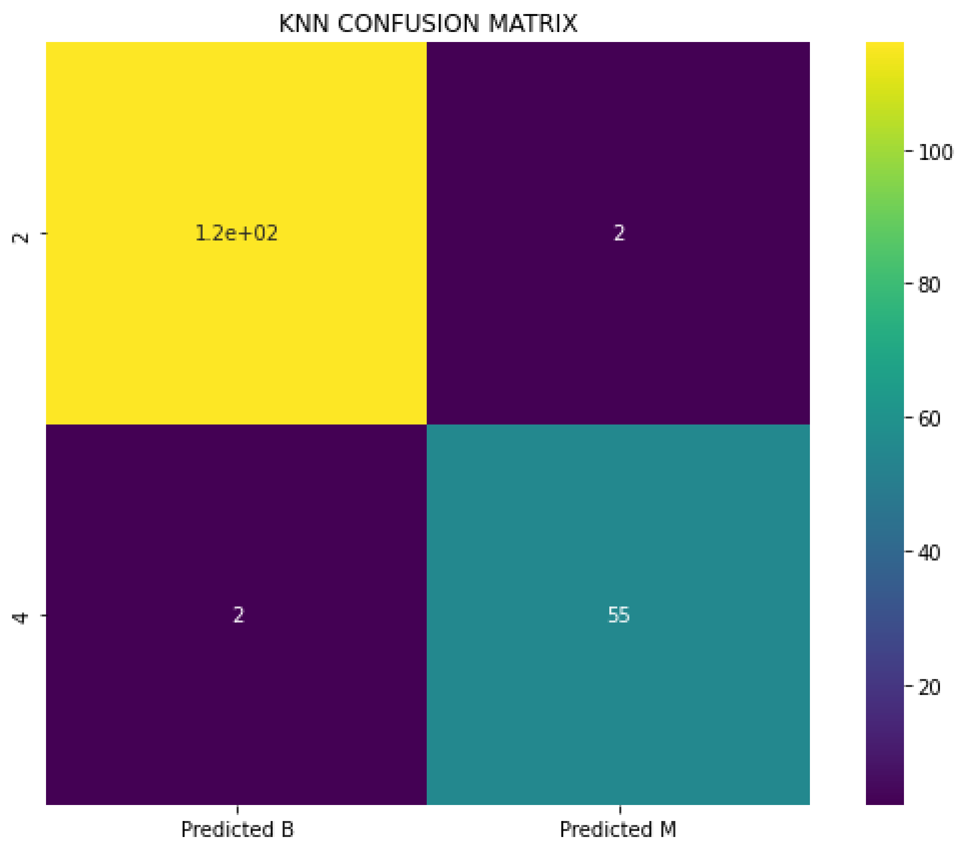
```
SVC Classification Report
............................................................
              precision    recall  f1-score   support

           2       0.96      0.97      0.97       118
           4       0.95      0.91      0.93        57

    accuracy                           0.95       175
   macro avg       0.95      0.94      0.95       175
weighted avg       0.95      0.95      0.95       175
```

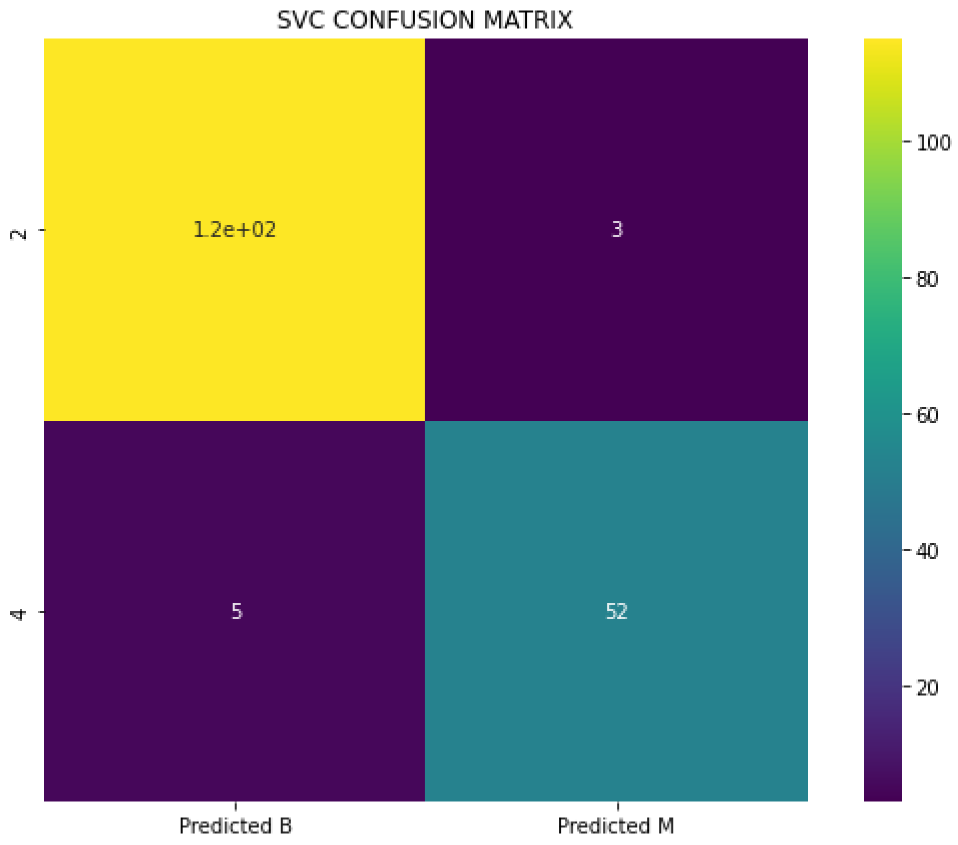
In [37]:

```python
knn_cm = pd.DataFrame(metrics.confusion_matrix(y_test, knn_preds),
                      index=[i for i in [2, 4]],
                      columns=[i for i in ['Predicted B', 'Predicted M']])
plt.figure(figsize=(10,7))
sns.heatmap(knn_cm, annot=True, cmap='viridis', square=True)
plt.title('KNN CONFUSION MATRIX')
plt.show()
```



In [38]:

```python
svc_cm = pd.DataFrame(metrics.confusion_matrix(y_test, svc_preds),
                      index=[i for i in [2, 4]],
                      columns=[i for i in ['Predicted B', 'Predicted M']])
plt.figure(figsize=(10,7))
sns.heatmap(svc_cm, annot=True, cmap='viridis', square=True)
plt.title('SVC CONFUSION MATRIX')
plt.show()
```


SVC CONFUSION MATRIX
2
4
1.2e+02
3
5
52
Predicted B
Predicted M
100
80
60
40
20

```
In [ ]:
```