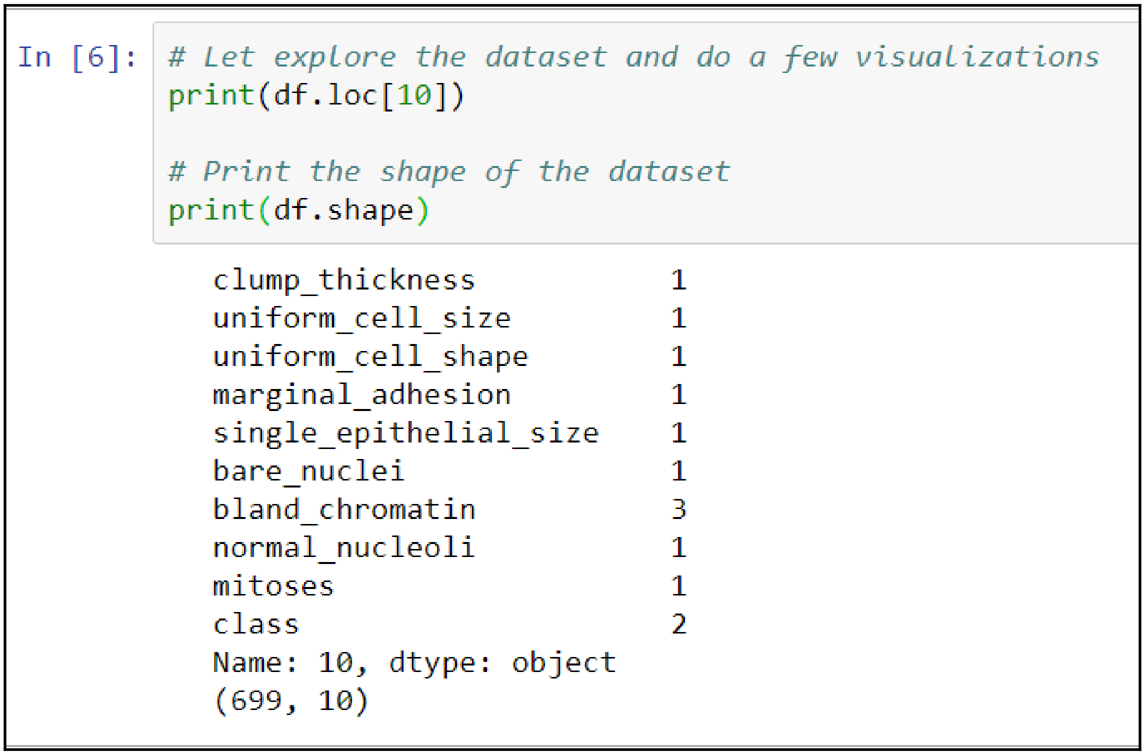
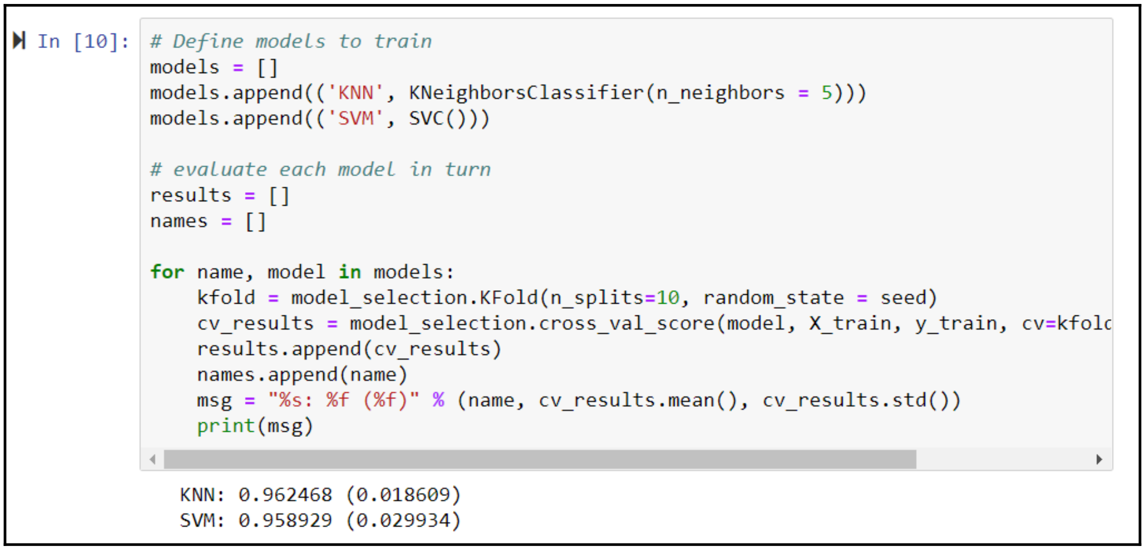
Chapter 1: Breast Cancer Detection

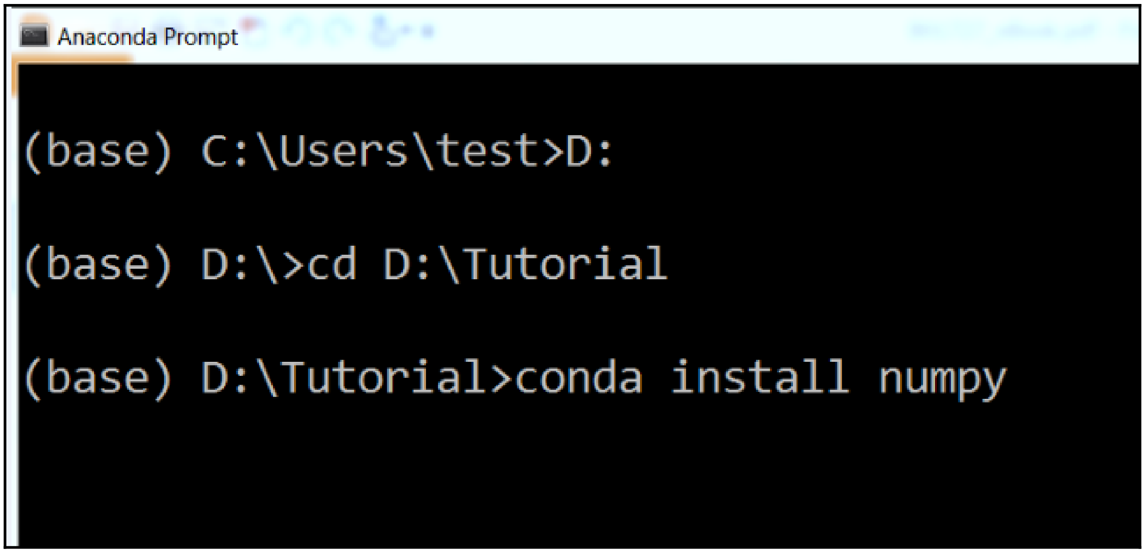


```
In [6]: # Let explore the dataset and do a few visualizations
        print(df.loc[10])

        # Print the shape of the dataset
        print(df.shape)
```

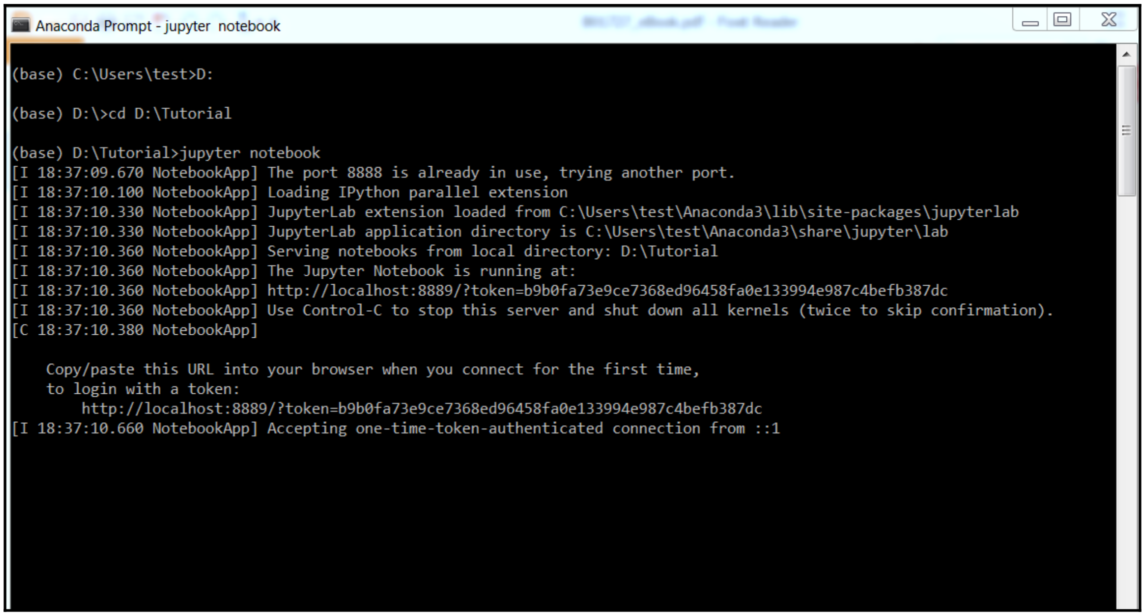
```
clump_thickness           1
uniform_cell_size         1
uniform_cell_shape        1
marginal_adhesion         1
single_epithelial_size    1
bare_nuclei               1
bland_chromatin           3
normal_nucleoli           1
mitoses                   1
class                     2
Name: 10, dtype: object
(699, 10)
```



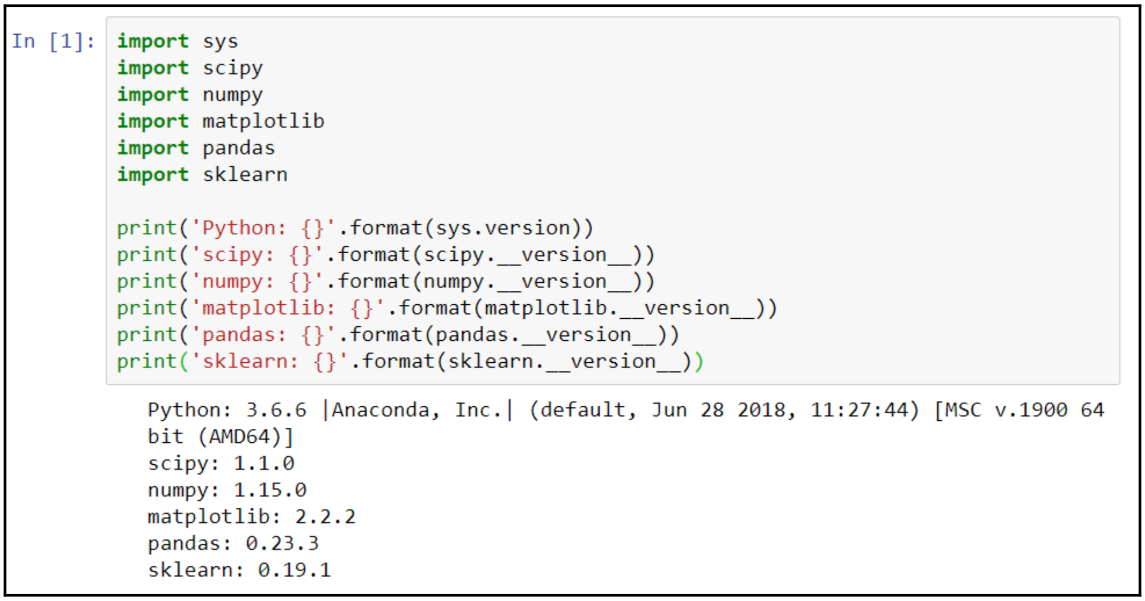
```
In [10]: # Define models to train
         models = []
         models.append(('KNN', KNeighborsClassifier(n_neighbors = 5)))
         models.append(('SVM', SVC()))

         # evaluate each model in turn
         results = []
         names = []

         for name, model in models:
             kfold = model_selection.KFold(n_splits=10, random_state = seed)
             cv_results = model_selection.cross_val_score(model, X_train, y_train, cv=kfolc
             results.append(cv_results)
             names.append(name)
             msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
             print(msg)

    KNN: 0.962468 (0.018609)
    SVM: 0.958929 (0.029934)
```



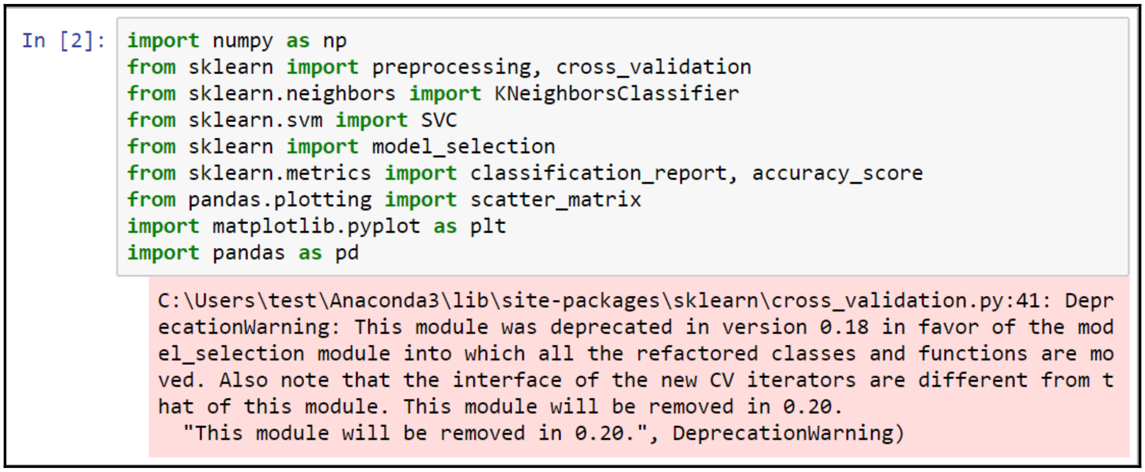
```
(base) C:\Users\test>D:

(base) D:\>cd D:\Tutorial

(base) D:\Tutorial>conda install numpy
```



```
(base) C:\Users\test>D:

(base) D:\>cd D:\Tutorial

(base) D:\Tutorial>jupyter notebook
[I 18:37:09.670 NotebookApp] The port 8888 is already in use, trying another port.
[I 18:37:10.100 NotebookApp] Loading IPython parallel extension
[I 18:37:10.330 NotebookApp] JupyterLab extension loaded from C:\Users\test\Anaconda3\lib\site-packages\jupyterlab
[I 18:37:10.330 NotebookApp] JupyterLab application directory is C:\Users\test\Anaconda3\share\jupyter\lab
[I 18:37:10.360 NotebookApp] Serving notebooks from local directory: D:\Tutorial
[I 18:37:10.360 NotebookApp] The Jupyter Notebook is running at:
[I 18:37:10.360 NotebookApp] http://localhost:8889/?token=b9b0fa73e9ce7368ed96458fa0e133994e987c4befb387dc
[I 18:37:10.360 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 18:37:10.380 NotebookApp]

    Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
        http://localhost:8889/?token=b9b0fa73e9ce7368ed96458fa0e133994e987c4befb387dc
[I 18:37:10.660 NotebookApp] Accepting one-time-token-authenticated connection from ::1
```

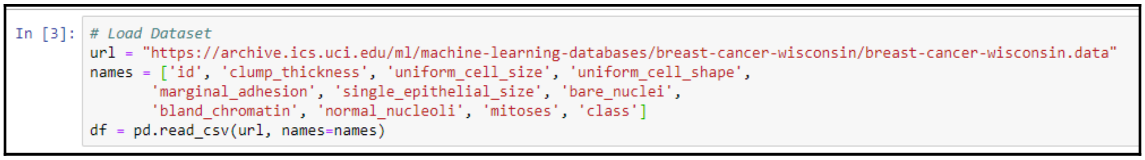


```
In [1]: import sys
        import scipy
        import numpy
        import matplotlib
        import pandas
        import sklearn

        print('Python: {}'.format(sys.version))
        print('scipy: {}'.format(scipy.__version__))
        print('numpy: {}'.format(numpy.__version__))
        print('matplotlib: {}'.format(matplotlib.__version__))
        print('pandas: {}'.format(pandas.__version__))
        print('sklearn: {}'.format(sklearn.__version__))

          Python: 3.6.6 |Anaconda, Inc.| (default, Jun 28 2018, 11:27:44) [MSC v.1900 64
          bit (AMD64)]
          scipy: 1.1.0
          numpy: 1.15.0
          matplotlib: 2.2.2
          pandas: 0.23.3
          sklearn: 0.19.1
```

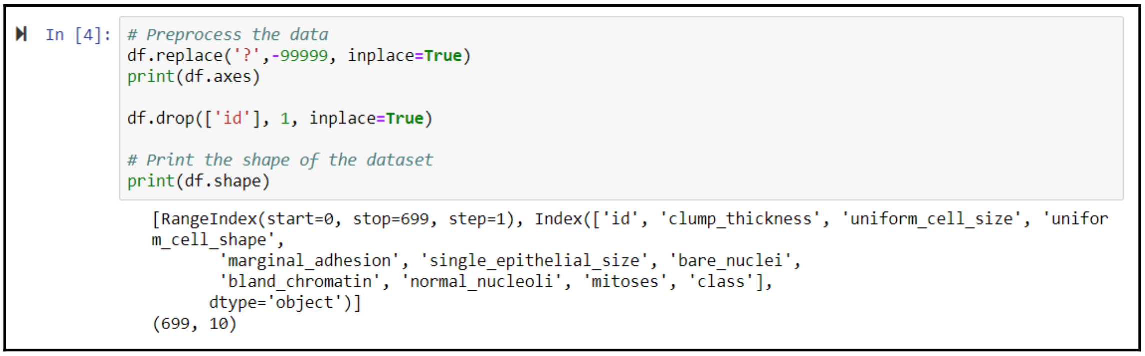


```
In [2]: import numpy as np
        from sklearn import preprocessing, cross_validation
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.svm import SVC
        from sklearn import model_selection
        from sklearn.metrics import classification_report, accuracy_score
        from pandas.plotting import scatter_matrix
        import matplotlib.pyplot as plt
        import pandas as pd
```

```
C:\Users\test\Anaconda3\lib\site-packages\sklearn\cross_validation.py:41: Depr
ecationWarning: This module was deprecated in version 0.18 in favor of the mod
el_selection module into which all the refactored classes and functions are mo
ved. Also note that the interface of the new CV iterators are different from t
hat of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

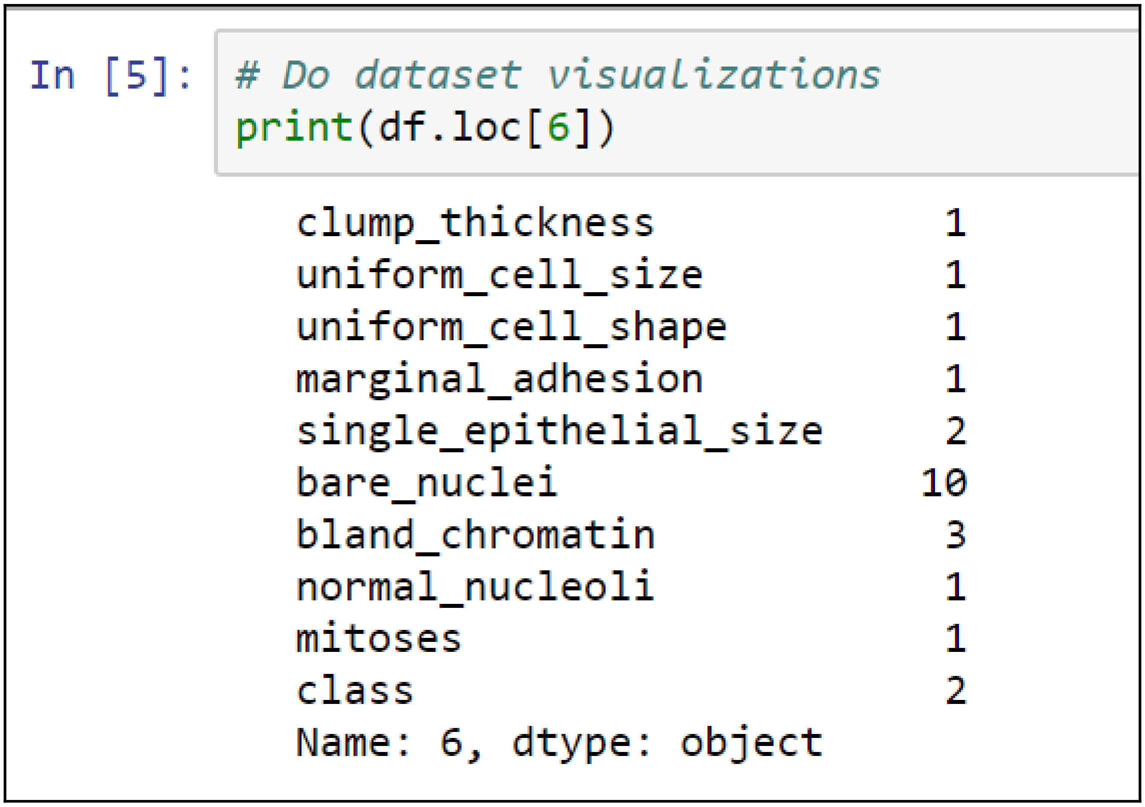


```
In [3]: # Load Dataset
        url = "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data"
        names = ['id', 'clump_thickness', 'uniform_cell_size', 'uniform_cell_shape',
               'marginal_adhesion', 'single_epithelial_size', 'bare_nuclei',
               'bland_chromatin', 'normal_nucleoli', 'mitoses', 'class']
        df = pd.read_csv(url, names=names)
```

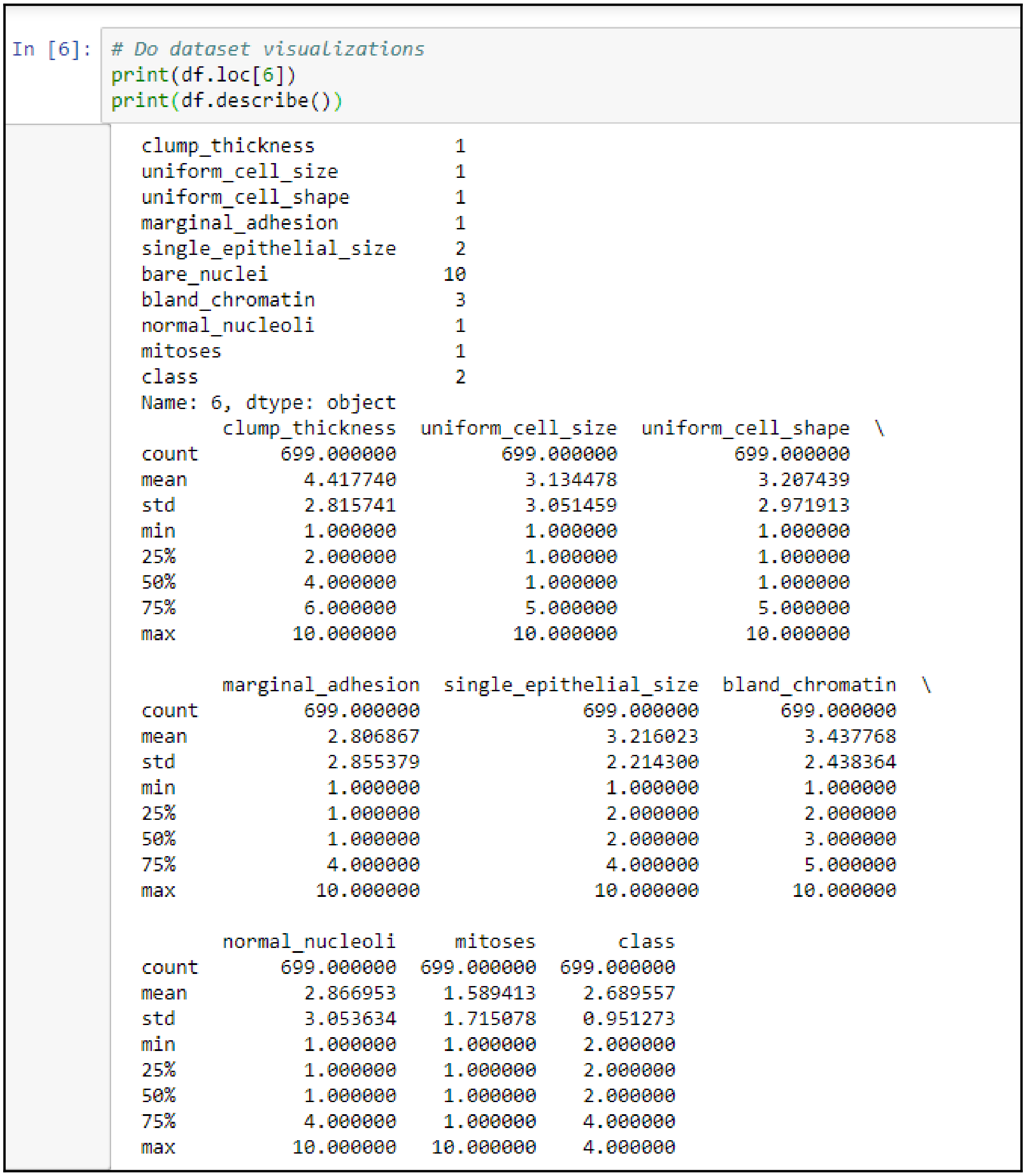


```
In [4]: # Preprocess the data
        df.replace('?',-99999, inplace=True)
        print(df.axes)

        df.drop(['id'], 1, inplace=True)

        # Print the shape of the dataset
        print(df.shape)
```

```
[RangeIndex(start=0, stop=699, step=1), Index(['id', 'clump_thickness', 'uniform_cell_size', 'unifor
m_cell_shape',
       'marginal_adhesion', 'single_epithelial_size', 'bare_nuclei',
       'bland_chromatin', 'normal_nucleoli', 'mitoses', 'class'],
      dtype='object')]
(699, 10)
```



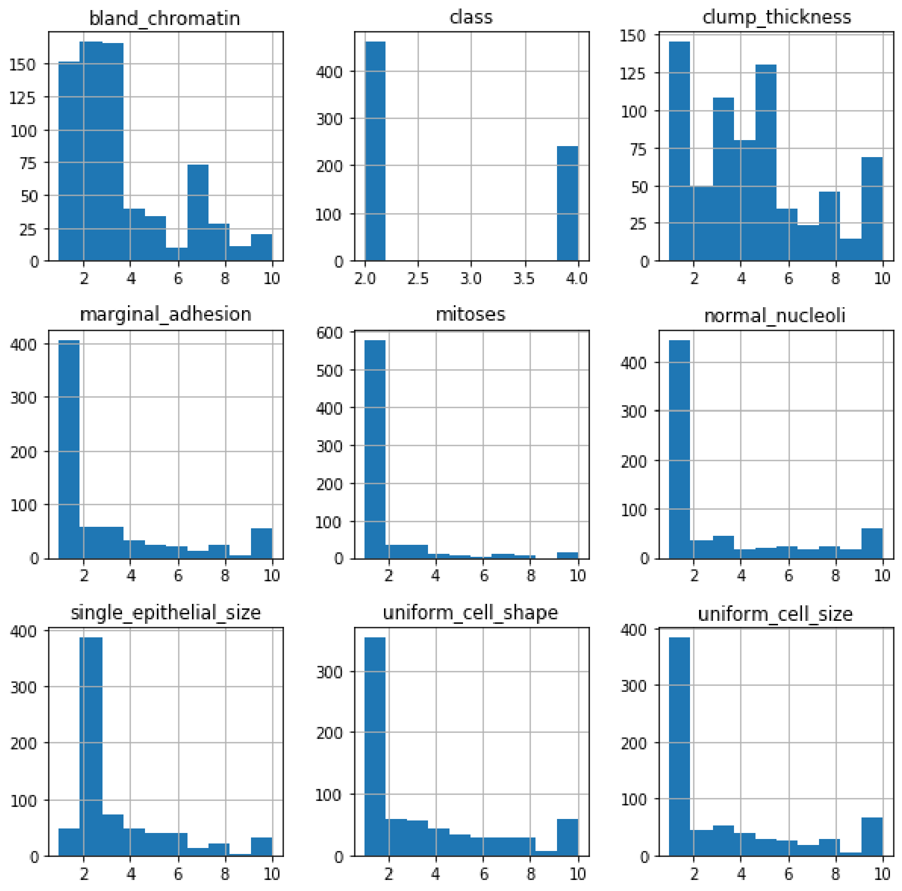
```
In [5]: # Do dataset visualizations
        print(df.loc[6])

clump_thickness             1
uniform_cell_size           1
uniform_cell_shape          1
marginal_adhesion           1
single_epithelial_size      2
bare_nuclei                10
bland_chromatin             3
normal_nucleoli             1
mitoses                     1
class                       2
Name: 6, dtype: object
```



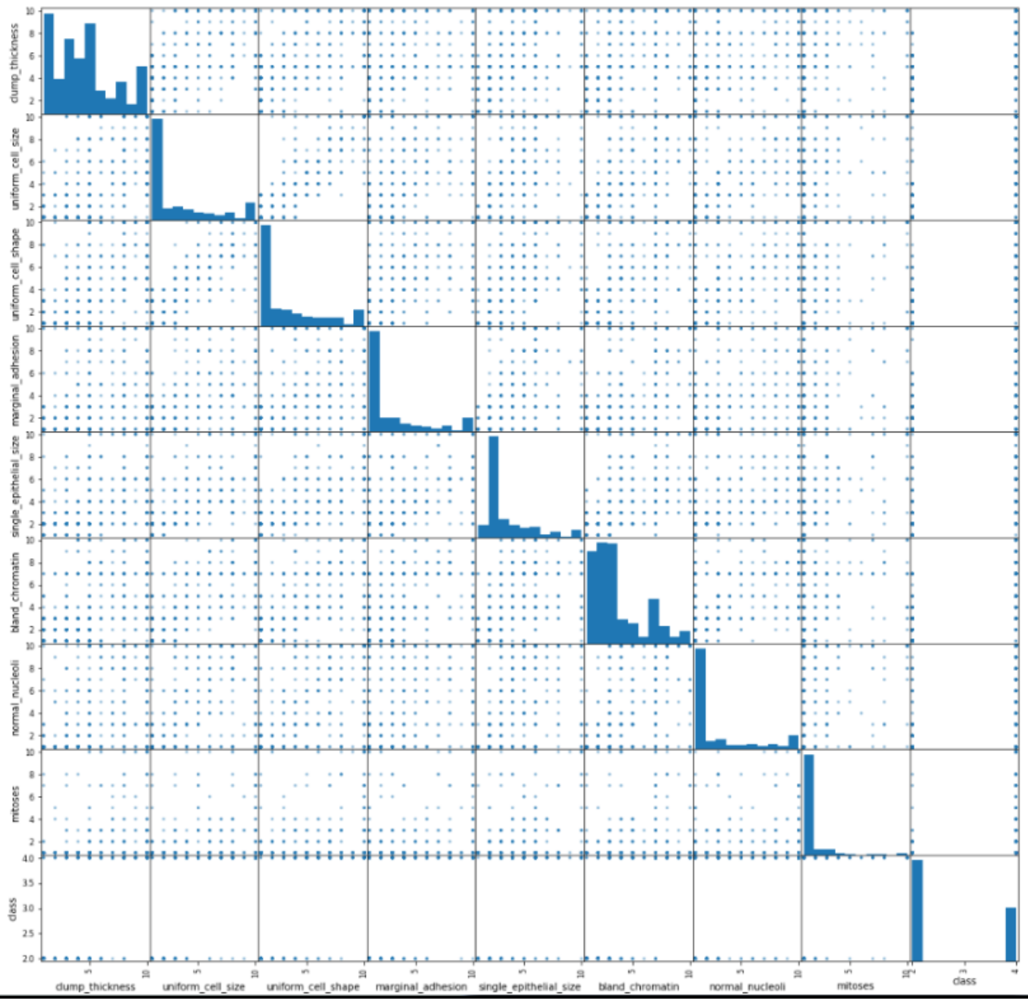
```
In [6]: # Do dataset visualizations
        print(df.loc[6])
        print(df.describe())
```

```
clump_thickness           1
uniform_cell_size         1
uniform_cell_shape        1
marginal_adhesion         1
single_epithelial_size    2
bare_nuclei              10
bland_chromatin           3
normal_nucleoli           1
mitoses                   1
class                     2
Name: 6, dtype: object
       clump_thickness  uniform_cell_size  uniform_cell_shape  \
count       699.000000         699.000000          699.000000
mean          4.417740           3.134478            3.207439
std           2.815741           3.051459            2.971913
min           1.000000           1.000000            1.000000
25%           2.000000           1.000000            1.000000
50%           4.000000           1.000000            1.000000
75%           6.000000           5.000000            5.000000
max          10.000000          10.000000           10.000000

       marginal_adhesion  single_epithelial_size  bland_chromatin  \
count         699.000000              699.000000       699.000000
mean            2.806867                3.216023         3.437768
std             2.855379                2.214300         2.438364
min             1.000000                1.000000         1.000000
25%             1.000000                2.000000         2.000000
50%             1.000000                2.000000         3.000000
75%             4.000000                4.000000         5.000000
max            10.000000               10.000000        10.000000

       normal_nucleoli     mitoses       class
count       699.000000  699.000000  699.000000
mean          2.866953    1.589413    2.689557
std           3.053634    1.715078    0.951273
min           1.000000    1.000000    2.000000
25%           1.000000    1.000000    2.000000
50%           1.000000    1.000000    2.000000
75%           4.000000    1.000000    4.000000
max          10.000000   10.000000    4.000000
```

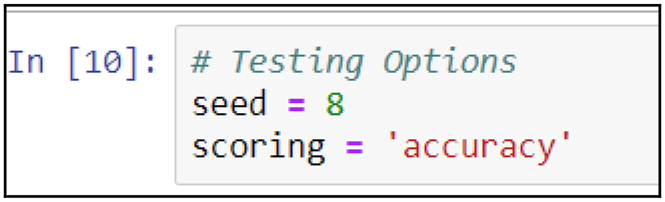
```
In [7]: # Plot histograms for each variable
        df.hist(figsize = (10, 10))
        plt.show()
```



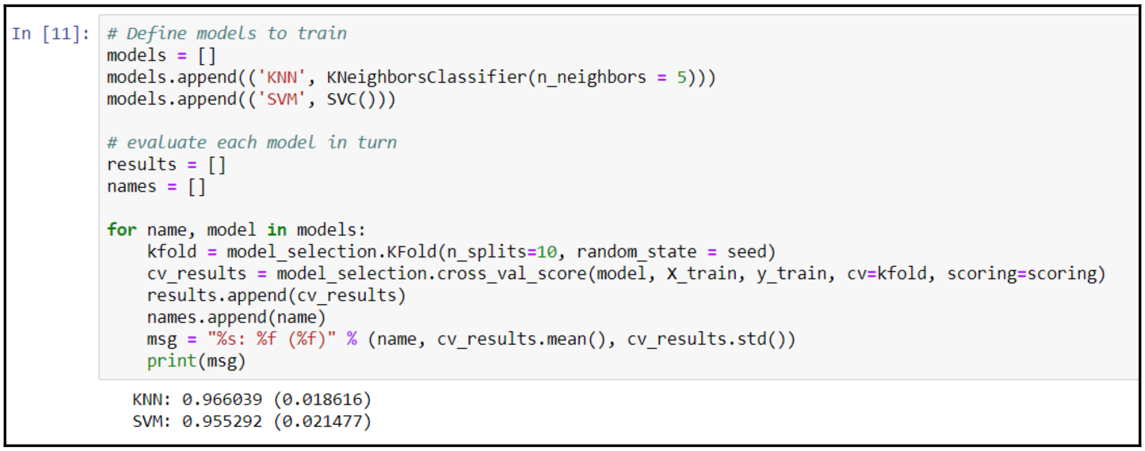
```
In [8]: # Create scatter plot matrix
        scatter_matrix(df, figsize = (18,18))
        plt.show()
```



```
In [9]: # Create X and Y datasets for training
        X = np.array(df.drop(['class'], 1))
        y = np.array(df['class'])

        X_train, X_test, y_train, y_test = cross_validation.train_test_split(X, y, test_size=0.2)
```

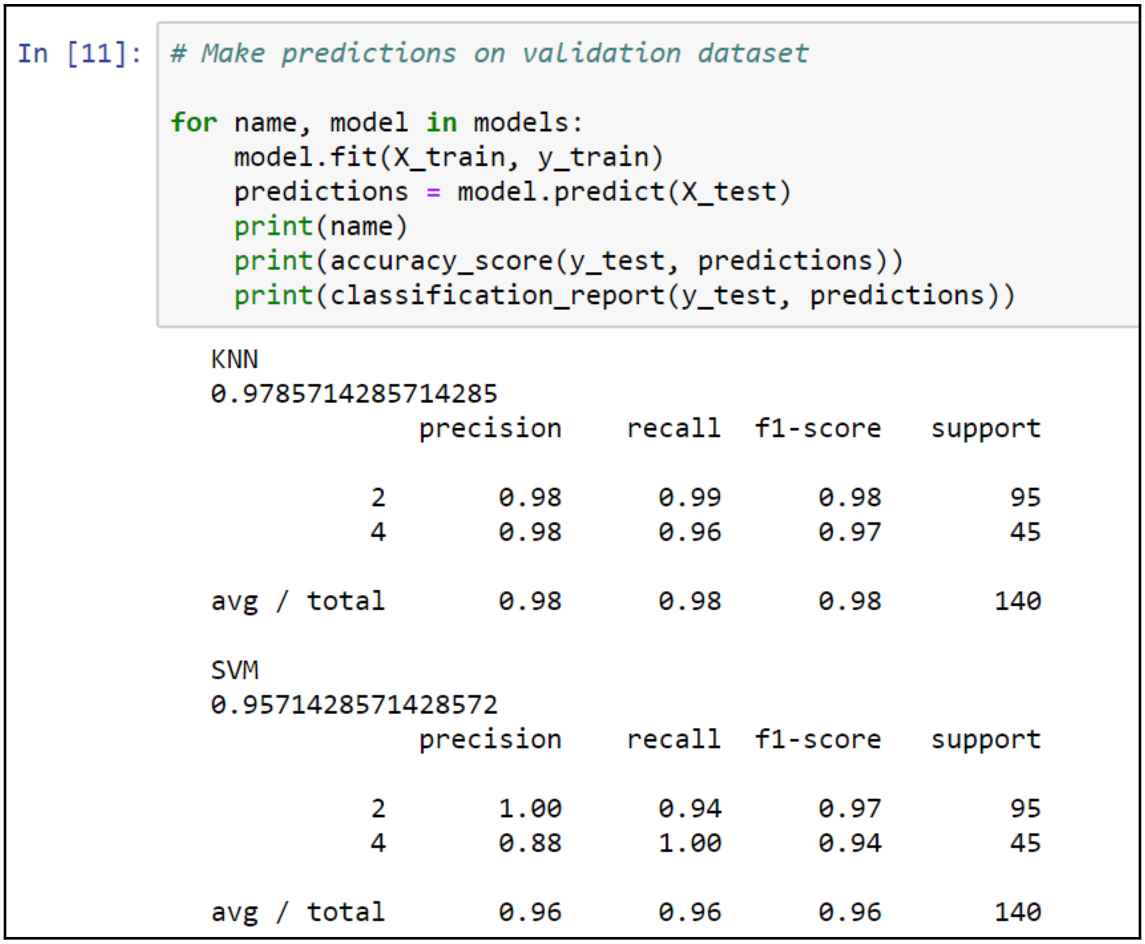


```
In [10]: # Testing Options
         seed = 8
         scoring = 'accuracy'
```



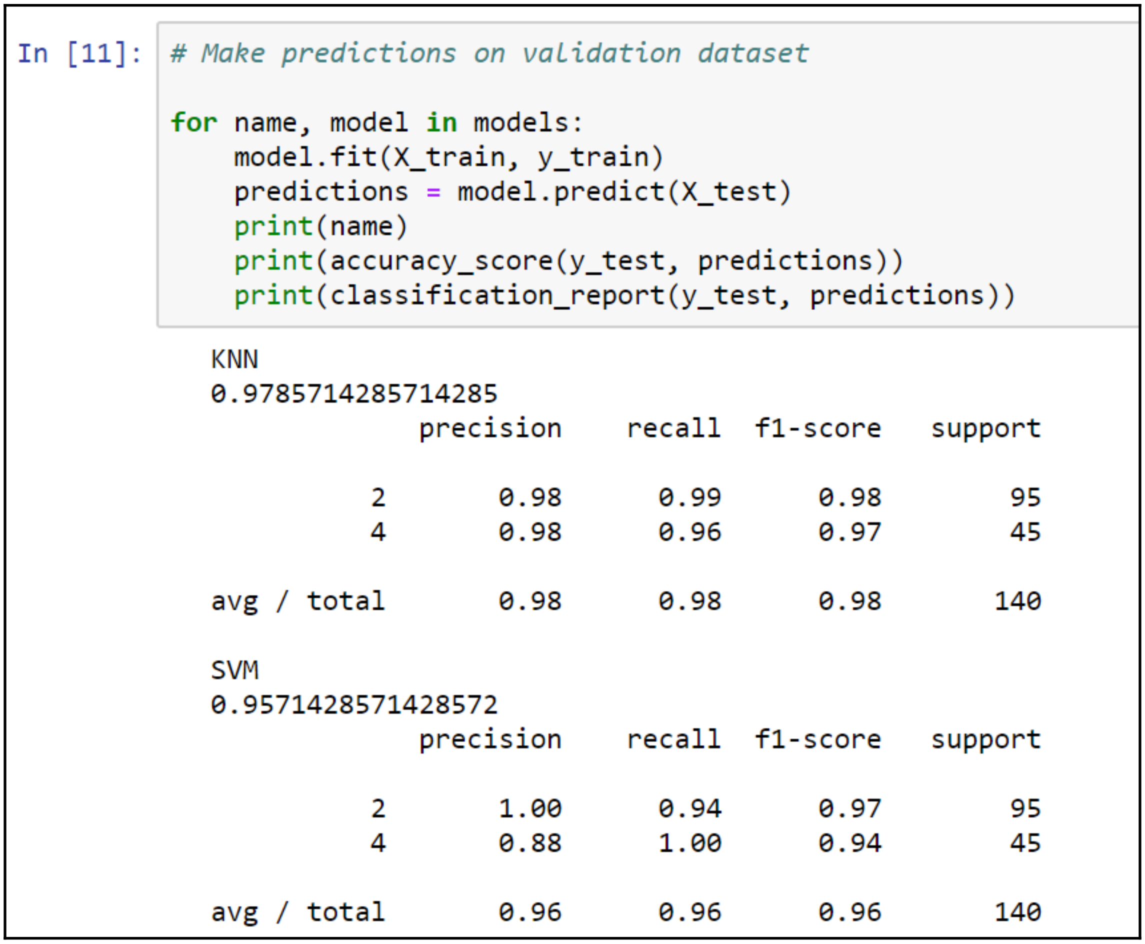
```
In [11]: # Define models to train
         models = []
         models.append(('KNN', KNeighborsClassifier(n_neighbors = 5)))
         models.append(('SVM', SVC()))

         # evaluate each model in turn
         results = []
         names = []

         for name, model in models:
             kfold = model_selection.KFold(n_splits=10, random_state = seed)
             cv_results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold, scoring=scoring)
             results.append(cv_results)
             names.append(name)
             msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
             print(msg)

            KNN: 0.966039 (0.018616)
            SVM: 0.955292 (0.021477)
```

```
In [11]: # Define models to train
         models = []
         models.append(('KNN', KNeighborsClassifier(n_neighbors = 5)))
         models.append(('SVM', SVC()))

         # evaluate each model in turn
         results = []
         names = []

         for name, model in models:
             kfold = model_selection.KFold(n_splits=10, random_state = seed)
             cv_results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold, scoring=scoring)
             results.append(cv_results)
             names.append(name)
             msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
             print(msg)

            KNN: 0.966039 (0.018616)
            SVM: 0.955292 (0.021477)
```



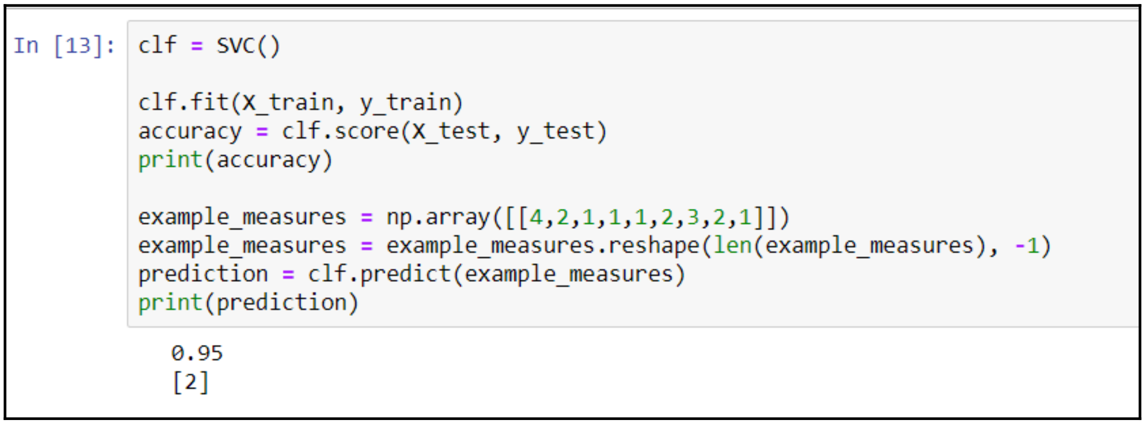
```
In [11]: # Make predictions on validation dataset

         for name, model in models:
             model.fit(X_train, y_train)
             predictions = model.predict(X_test)
             print(name)
             print(accuracy_score(y_test, predictions))
             print(classification_report(y_test, predictions))
```

```
KNN
0.9785714285714285
             precision    recall  f1-score   support

          2       0.98      0.99      0.98        95
          4       0.98      0.96      0.97        45

avg / total       0.98      0.98      0.98       140

SVM
0.9571428571428572
             precision    recall  f1-score   support

          2       1.00      0.94      0.97        95
          4       0.88      1.00      0.94        45

avg / total       0.96      0.96      0.96       140
```



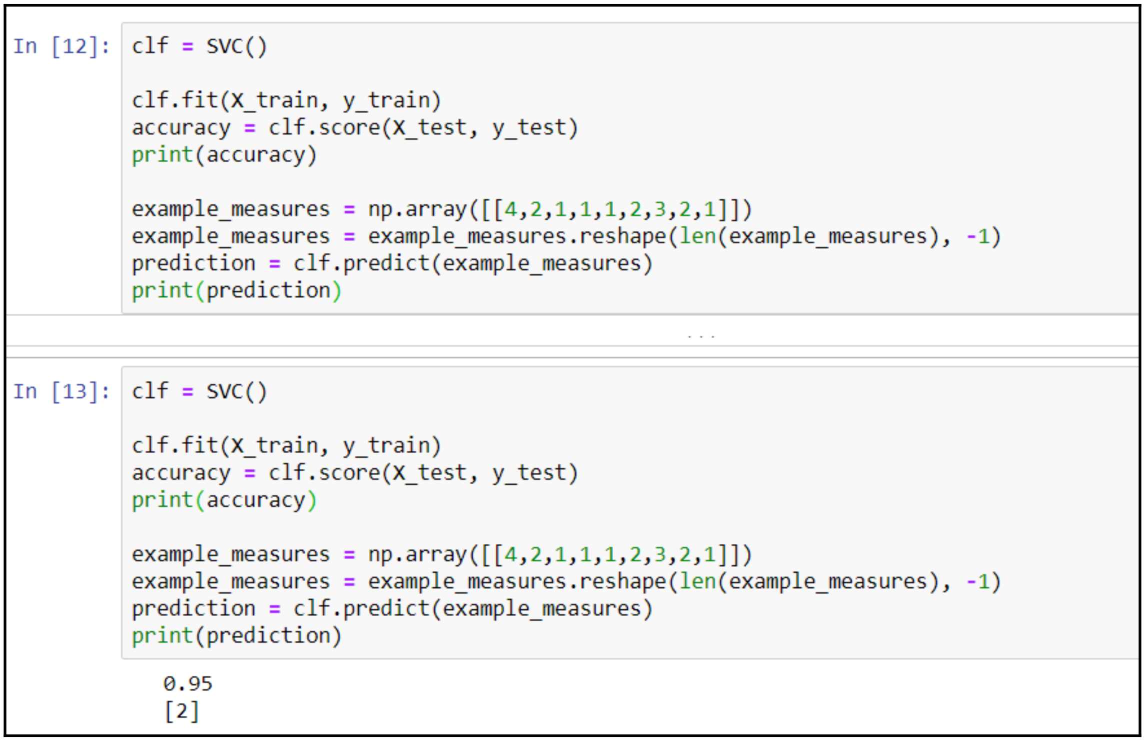
```
In [11]: # Make predictions on validation dataset

         for name, model in models:
             model.fit(X_train, y_train)
             predictions = model.predict(X_test)
             print(name)
             print(accuracy_score(y_test, predictions))
             print(classification_report(y_test, predictions))
```

```
KNN
0.9785714285714285
             precision    recall  f1-score   support

          2       0.98      0.99      0.98        95
          4       0.98      0.96      0.97        45

avg / total       0.98      0.98      0.98       140

SVM
0.9571428571428572
             precision    recall  f1-score   support

          2       1.00      0.94      0.97        95
          4       0.88      1.00      0.94        45

avg / total       0.96      0.96      0.96       140
```



```
In [13]: clf = SVC()

         clf.fit(X_train, y_train)
         accuracy = clf.score(X_test, y_test)
         print(accuracy)

         example_measures = np.array([[4,2,1,1,1,2,3,2,1]])
         example_measures = example_measures.reshape(len(example_measures), -1)
         prediction = clf.predict(example_measures)
         print(prediction)
```

```
0.95
[2]
```



```
In [12]: clf = SVC()

         clf.fit(X_train, y_train)
         accuracy = clf.score(X_test, y_test)
         print(accuracy)

         example_measures = np.array([[4,2,1,1,1,2,3,2,1]])
         example_measures = example_measures.reshape(len(example_measures), -1)
         prediction = clf.predict(example_measures)
         print(prediction)
```

```
In [13]: clf = SVC()

         clf.fit(X_train, y_train)
         accuracy = clf.score(X_test, y_test)
         print(accuracy)

         example_measures = np.array([[4,2,1,1,1,2,3,2,1]])
         example_measures = example_measures.reshape(len(example_measures), -1)
         prediction = clf.predict(example_measures)
         print(prediction)
```

```
0.95
[2]
```