```python
from sklearn.ensemble import RandomForestClassifier
rfr1 = RandomForestClassifier().fit(x_os,y_os.values.ravel())
y_pred = rfr1.predict(x_test_os)

rfr1 = RandomForestClassifier()
```

```python
rfr1.fit(x_os, y_os.values.ravel())
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```python
y_pred = rfr1.predict(x_test_os)
```

```python
y_pred = rfr1.predict(x_test_os)
```

```python
print(classification_report(y_test_os,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       122
           1       0.76      0.90      0.83       122
           2       0.91      0.98      0.94       122
           3       0.78      0.83      0.80       122
           4       0.46      0.92      0.62       122
           5       0.75      0.70      0.73       122
           6       0.63      0.48      0.54       122

    accuracy                           0.69       854
   macro avg       0.61      0.69      0.64       854
weighted avg       0.61      0.69      0.64       854
```

```python
train_score = accuracy_score(y_os, rfr1.predict(x_os))
train_score
```

```
1.0
```

```
from xgboost import XGBClassifier
xgb1 = XGBClassifier()
xgb1.fit(x_os,y_os)
```

```
XGBClassifier
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, objective='multi:softprob',
              predictor='auto', random_state=0, reg_alpha=0, ...)
```

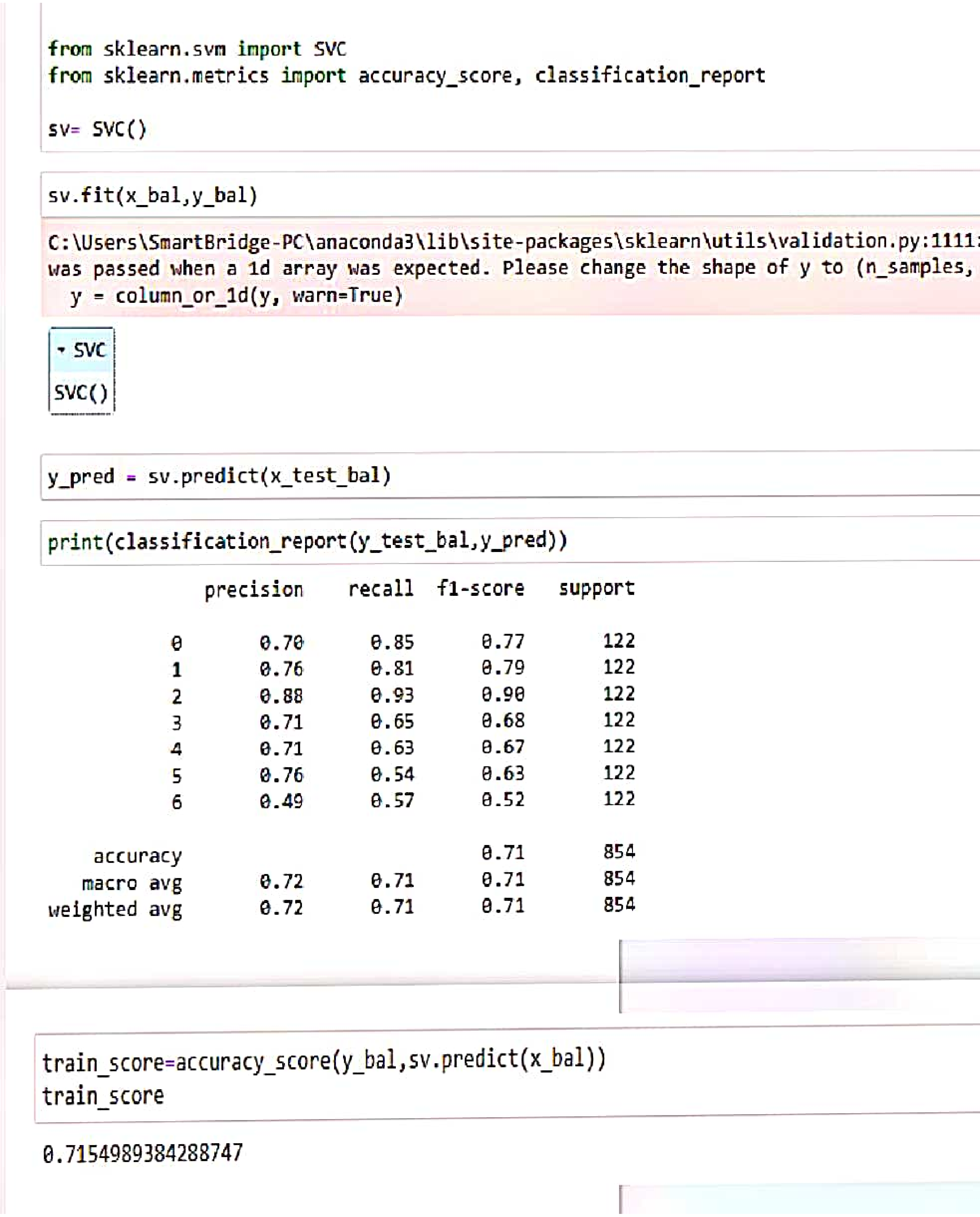
```
y_pred = xgb1.predict(x_test_os)
```

```
print(classification_report(y_test_os,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.70      0.13      0.22       122
           1       0.75      0.93      0.84       122
           2       0.95      0.99      0.97       122
           3       0.76      0.77      0.77       122
           4       0.48      0.85      0.61       122
           5       0.79      0.71      0.75       122
           6       0.62      0.52      0.57       122

    accuracy                           0.70       854
   macro avg       0.72      0.70      0.67       854
weighted avg       0.72      0.70      0.67       854
```

```
accuracy_score(y_test_os,y_pred)
```

```
0.7014051522248244
```



```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

sv= SVC()
```

```
sv.fit(x_bal,y_bal)
```

```
C:\Users\SmartBridge-PC\anaconda3\lib\site-packages\sklearn\utils\validation.py:1111:
was passed when a 1d array was expected. Please change the shape of y to (n_samples,
  y = column_or_1d(y, warn=True)
```

```
SVC()
```

```
y_pred = sv.predict(x_test_bal)
```

```
print(classification_report(y_test_bal,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.70      0.85      0.77       122
           1       0.76      0.81      0.79       122
           2       0.88      0.93      0.90       122
           3       0.71      0.65      0.68       122
           4       0.71      0.63      0.67       122
           5       0.76      0.54      0.63       122
           6       0.49      0.57      0.52       122

    accuracy                           0.71       854
   macro avg       0.72      0.71      0.71       854
weighted avg       0.72      0.71      0.71       854
```

```
train_score=accuracy_score(y_bal,sv.predict(x_bal))
train_score
```

```
0.7154989384288747
```

```
In [68]: model = Sequential()
```

```
In [69]: model.add(Dense(units = 128, activation='relu', input_shape=(10,)))
```

```
In [70]: model.add(Dense(units = 128, activation='relu', kernel_initializer='random
         model.add(Dropout(0.2))
         model.add(Dense(units = 256, activation='relu', kernel_initializer='random
         model.add(Dropout(0.2))
         model.add(Dense(units = 128, activation='relu', kernel_initializer='random
```

```
In [71]: model.add(Dense(units = 1, activation='sigmoid'))
```

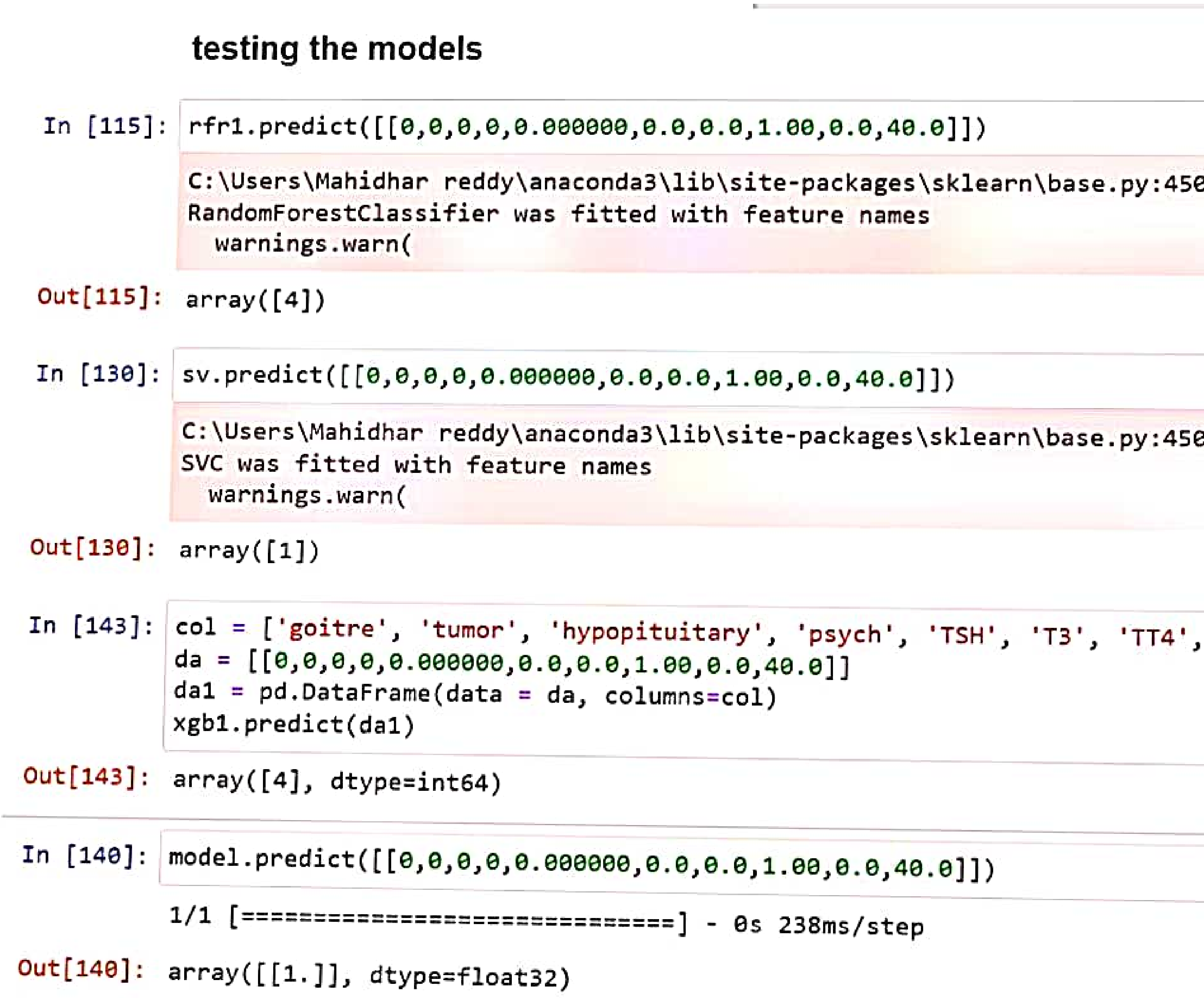
```
In [72]: model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               1408

 dense_1 (Dense)             (None, 128)               16512

 dropout (Dropout)           (None, 128)               0

 dense_2 (Dense)             (None, 256)               33024

 dropout_1 (Dropout)         (None, 256)               0

 dense_3 (Dense)             (None, 128)               32896

 dense_4 (Dense)             (None, 1)                 129

=================================================================
Total params: 83,969
Trainable params: 83,969
Non-trainable params: 0
_________________________________________________________________
```

```
In [73]: model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc
```

```
In [75]: model.fit(x_bal,y_bal, validation_data=[x_test_bal, y_test_bal], epochs=1
```

```
Epoch 1/15
104/104 [==============================] - 9s 15ms/step - loss: -18416.0
_accuracy: 0.1429
Epoch 2/15
104/104 [==============================] - 1s 8ms/step - loss: -2626274.
val_accuracy: 0.1429
Epoch 3/15
104/104 [==============================] - 1s 9ms/step - loss: -42823204.
- val_accuracy: 0.1429
Epoch 4/15
104/104 [==============================] - 1s 9ms/step - loss: -27723212
- val_accuracy: 0.1429
Epoch 5/15
104/104 [==============================] - 1s 8ms/step - loss: -10978827
00 - val_accuracy: 0.1429
Epoch 6/15
104/104 [==============================] - 1s 8ms/step - loss: -32085196
```



testing the models

```
In [115]: rfr1.predict([[0,0,0,0,0.000000,0.0,0.0,1.00,0.0,40.0]])

          C:\Users\Mahidhar reddy\anaconda3\lib\site-packages\sklearn\base.py:450
          RandomForestClassifier was fitted with feature names
            warnings.warn(

Out[115]: array([4])

In [130]: sv.predict([[0,0,0,0,0.000000,0.0,0.0,1.00,0.0,40.0]])

          C:\Users\Mahidhar reddy\anaconda3\lib\site-packages\sklearn\base.py:450
          SVC was fitted with feature names
            warnings.warn(

Out[130]: array([1])

In [143]: col = ['goitre', 'tumor', 'hypopituitary', 'psych', 'TSH', 'T3', 'TT4',
          da = [[0,0,0,0,0.000000,0.0,0.0,1.00,0.0,40.0]]
          da1 = pd.DataFrame(data = da, columns=col)
          xgb1.predict(da1)

Out[143]: array([4], dtype=int64)

In [140]: model.predict([[0,0,0,0,0.000000,0.0,0.0,1.00,0.0,40.0]])

          1/1 [==============================] - 0s 238ms/step

Out[140]: array([[1.]], dtype=float32)
```