Titanic Binary Classification

January 17, 2018

1 Importing the libraries

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
```

2 Importing the dataset

```
In [4]: X_train = pd.read_csv('Data/train_wrangled_X.csv').values
        y_train = pd.read_csv('Data/train_wrangled_y.csv').values
        X_test = pd.read_csv('Data/test_wrangled.csv').values
        y_test = pd.read_csv('Data/y_test_wrangled.csv').values
```

3 Feature Scaling

```
In [5]: from sklearn.preprocessing import StandardScaler
        sc = StandardScaler()
        X_train = sc.fit_transform(X_train)
        X_test = sc.transform(X_test)
```

4 Fitting Random Forest Classification to the Training set

```
In [6]: from sklearn.ensemble import RandomForestClassifier
        classifier = RandomForestClassifier(n_estimators = 10,
                                            criterion = 'entropy', random_state = 0)
        classifier.fit(X_train, y_train)

C:\Users\Ripti\Anaconda3\lib\site-packages\ipykernel_launcher.py:7: DataConversionWarning: A col
  import sys


Out[6]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
                    max_depth=None, max_features='auto', max_leaf_nodes=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=1, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
                    oob_score=False, random_state=0, verbose=0, warm_start=False)
```

5 Predicting the Test set results

```
In [7]: y_pred = classifier.predict(X_test)
```

6 Making the Confusion Matrix to Determine Accuracy

```
In [23]: from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(y_test, y_pred)
         print(cm)
         print("Accuracy: "+str(round((((
             241+114)/(241+114+38+25))*100),2))+"%")
```

```
[[241  25]
 [ 38 114]]
Accuracy: 84.93%
```

7 Using a Dense Nueral Network

```
In [24]: from keras.models import Sequential
         from keras.layers import Dense
```

```
C:\Users\Ripti\Anaconda3\lib\site-packages\h5py\__init__.py:34: FutureWarning: Conversion of the
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

8 Define the Model and Layers

```
In [38]: model = Sequential()
         model.add(Dense(12, input_dim=10, activation='relu'))
         model.add(Dense(10, activation='sigmoid'))
         model.add(Dense(10, activation='relu'))
         model.add(Dense(10, activation='softmax'))
         model.add(Dense(1, activation='sigmoid'))
         model.compile(loss='binary_crossentropy',
         optimizer='adam', metrics=['accuracy'])
```

9 Fit the model to the training data

```
In [41]: model.fit(X_train, y_train, epochs=20, batch_size=25)
```

```
Epoch 1/20
891/891 [==============================] - 0s - loss: 0.4269 - acc: 0.8384
Epoch 2/20
891/891 [==============================] - 0s - loss: 0.4247 - acc: 0.8406
```

```
Epoch 3/20
891/891 [==============================] - 0s - loss: 0.4230 - acc: 0.8440
Epoch 4/20
891/891 [==============================] - 0s - loss: 0.4220 - acc: 0.8429
Epoch 5/20
891/891 [==============================] - 0s - loss: 0.4216 - acc: 0.8451
Epoch 6/20
891/891 [==============================] - 0s - loss: 0.4200 - acc: 0.8429
Epoch 7/20
891/891 [==============================] - 0s - loss: 0.4209 - acc: 0.8406
Epoch 8/20
891/891 [==============================] - 0s - loss: 0.4186 - acc: 0.8429
Epoch 9/20
891/891 [==============================] - 0s - loss: 0.4188 - acc: 0.8451
Epoch 10/20
891/891 [==============================] - 0s - loss: 0.4165 - acc: 0.8418
Epoch 11/20
891/891 [==============================] - 0s - loss: 0.4168 - acc: 0.8474
Epoch 12/20
891/891 [==============================] - 0s - loss: 0.4154 - acc: 0.8429
Epoch 13/20
891/891 [==============================] - 0s - loss: 0.4149 - acc: 0.8440
Epoch 14/20
891/891 [==============================] - 0s - loss: 0.4136 - acc: 0.8451
Epoch 15/20
891/891 [==============================] - 0s - loss: 0.4131 - acc: 0.8440
Epoch 16/20
891/891 [==============================] - 0s - loss: 0.4125 - acc: 0.8474
Epoch 17/20
891/891 [==============================] - 0s - loss: 0.4117 - acc: 0.8462
Epoch 18/20
891/891 [==============================] - 0s - loss: 0.4110 - acc: 0.8440
Epoch 19/20
891/891 [==============================] - 0s - loss: 0.4117 - acc: 0.8451
Epoch 20/20
891/891 [==============================] - 0s - loss: 0.4109 - acc: 0.8418
```

```
Out[41]: <keras.callbacks.History at 0x1d26fbff320>
```

10 Evaluate the model against the test data

```
In [42]: scores = model.evaluate(X_test, y_test)
         print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

```
 32/418 [=>...] - ETA: 0s
acc: 90.19%
```

```
In [64]: y_pred = model.predict(X_test).round()
         np.savetxt("predictions.csv",y_pred,delimiter=",")
```