Artificial Neural Network Regression

Importing the libraries

In [0]:

```
import numpy as np
import pandas as pd
import tensorflow as tf
```

In [2]:

```
tf.__version__
```

Out[2]:

```
'2.2.0-rc2'
```

Part 1 - Data Preprocessing

Importing the dataset

In [0]:

```
dataset = pd.read_excel('Folds5x2_pp.xlsx')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

Splitting the dataset into the Training set and Test set

In [0]:

```
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, r
```

Part 2 - Building the ANN

Initializing the ANN

In [0]:

```
ann = tf.keras.models.Sequential()
```

Adding the input layer and the first hidden layer

In [0]:

```
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

Adding the second hidden layer

In [0]:

```
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

Adding the output layer

In [0]:
```
ann.add(tf.keras.layers.Dense(units=1))
```

Part 3 - Training the ANN

Compiling the ANN

In [0]:
```
ann.compile(optimizer = 'adam', loss = 'mean_squared_error')
```

Training the ANN model on the Training set

In [10]:
```
ann.fit(X_train, y_train, batch_size = 32, epochs = 100)
```

```
Epoch 1/100
240/240 [==============================] - 0s 2ms/step - loss: 82110.9922
Epoch 2/100
240/240 [==============================] - 0s 1ms/step - loss: 656.9040
Epoch 3/100
240/240 [==============================] - 0s 1ms/step - loss: 429.4050
Epoch 4/100
240/240 [==============================] - 0s 1ms/step - loss: 417.9655
Epoch 5/100
240/240 [==============================] - 0s 1ms/step - loss: 405.0805
Epoch 6/100
240/240 [==============================] - 0s 1ms/step - loss: 388.9652
Epoch 7/100
240/240 [==============================] - 0s 2ms/step - loss: 371.9583
Epoch 8/100
240/240 [==============================] - 0s 1ms/step - loss: 353.7564
Epoch 9/100
240/240 [==============================] - 0s 1ms/step - loss: 332.1931
Epoch 10/100
240/240 [==============================] - 0s 1ms/step - loss: 310.8850
```

Predicting the results of the Test set

In [11]:
```
y_pred = ann.predict(X_test)
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_te
```

```
[[430.79 431.23]
 [461.8  460.01]
 [465.29 461.14]
 ...
 [472.51 473.26]
 [439.39 438.  ]
 [458.55 463.28]]
```