Implementing Artificial Neural Network(Classification) in Python

Step 1: Importing necessary Libraries

```
import numpy as np

import pandas as pd

import tensorflow as tf

from sklearn.preprocessing import LabelEncoder

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from keras.models import Sequential

from keras.layers import Input, Dense, Activation, Dropout, BatchNormalization
```

Step 2: Loading Dataset

Step 3: Separation of Dependent and Independent Variable

X = Generating Matrix of Features (X)

y = Generating Dependent Variable Vector(Y)

Step 4: Encoding Categorical Variable

Step 5: Splitting Dataset into Training and Testing Dataset

Step 6: Performing Feature Scaling

Step 7: Initialising ANN

```
model = Sequential()
```

Creating input Layer

```
model.add(Input(shape = (4,)))
```

Creating Hidden Layers

```
model.add(Dense(units=10, activation="relu"))
model.add(Dense(units=5, activation="relu"))
.....
```

Creating Output Layer

```
# for binary classification problem
model.add(Dense(units=1,  activation="sigmoid"))

# for multiclass classification problem
model.add(Dense(units=3,  activation="softmax"))
```

Step 8: Compiling Artificial Neural Network

```
# Compiling ANN for binary classification problem
model.compile(optimizer="adam", loss="binary_crossentropy",metrics=['accuracy'])

# Compiling ANN for multiclass classification problem
model.compile(optimizer = opt, loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

where opt = tf.keras.optimizers.SGD(learning_rate = 0.1)

Summarize the Model

```
model.summary()
```

Step 9: Fitting Artificial Neural Network

```
# Fitting ANN
model.fit(X_train, Y_train, batch_size=32, epochs = 100)
```

Step 9: Hyperparameter Tuning

```
model1 = Sequential()
model1.add(Input(shape = (4,)))  # for input layer
model1.add(Dense(10, activation="relu", kernel_initializer="he_normal"))
model1.add(BatchNormalization())
model1.add(Dense(20, activation="relu", kernel_initializer="he_normal"))
```

```
model1.add(Dropout(0.4)) # preferred value <= 0.5
model1.add(Dense(3,  activation="softmax"))    # for output layer
```

Compile the Model

```
model1.compile(optimizer = opt, loss = 'categorical_crossentropy', metrics = ['accuracy'])
model1.summary()
```

Step 10: Make predictions

```
pred = model1.predict(X_test)
pred[0]
```

Compare the prediction with actual labels

Step 11: Classification Report

```
from sklearn.metrics import classification_report
actuals = np.argmax(y_test, axis = 1)
predictions = np.argmax(pred, axis = 1)
print(classification_report(actuals, predictions))
```

Step 11: Write Interpretation about the result