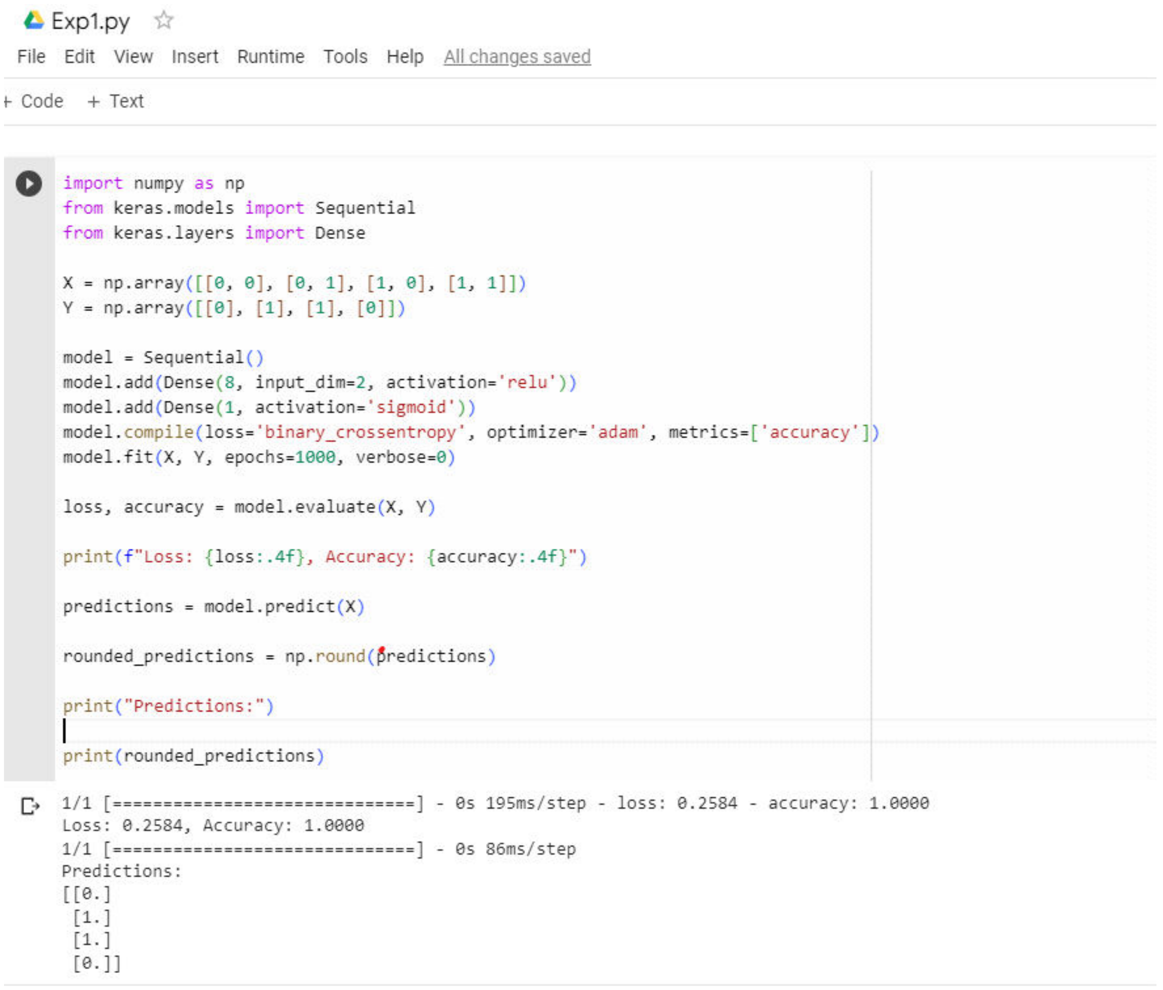
Exp 1

Implementation of perceptron using Tensorflow and Keras

Code:-

```
import numpy as np

from keras.models import Sequential

from keras.layers import Dense

X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])

Y = np.array([[0], [1], [1], [0]])

model = Sequential()

model.add(Dense(8, input_dim=2, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

model.fit(X, Y, epochs=1000, verbose=0)

loss, accuracy = model.evaluate(X, Y)

print(f"Loss: {loss:.4f}, Accuracy: {accuracy:.4f}")

predictions = model.predict(X)

rounded_predictions = np.round(predictions)

print("Predictions:")

print(rounded_predictions)
```

Output:-



```python
import numpy as np
from keras.models import Sequential
from keras.layers import Dense

X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
Y = np.array([[0], [1], [1], [0]])

model = Sequential()
model.add(Dense(8, input_dim=2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X, Y, epochs=1000, verbose=0)

loss, accuracy = model.evaluate(X, Y)

print(f"Loss: {loss:.4f}, Accuracy: {accuracy:.4f}")

predictions = model.predict(X)

rounded_predictions = np.round(predictions)

print("Predictions:")

print(rounded_predictions)
```

```
1/1 [==============================] - 0s 195ms/step - loss: 0.2584 - accuracy: 1.0000
Loss: 0.2584, Accuracy: 1.0000
1/1 [==============================] - 0s 86ms/step
Predictions:
[[0.]
 [1.]
 [1.]
 [0.]]
```